Compressão

Prof. Jose J. Camata
Prof. Marcelo Caniato
camata@ice.ufjf.br
marcelo.caniato@ice.ufjf.br

Tópicos

1. Introdução
2. Compressão RLE
3. Compressão de Huffman
4. Huffman adaptativo

Introdução

- O que fazemos quando queremos comunicar alguma coisa de forma rápida?
  - Sabe aquela história que a moça perde o sapato?
  - Aquela série que tem 6 amigos que moram no mesmo prédio?

Na vida naturalmente comprimimos informações

- Para comprimir informações de forma geral precisamos conhecer alguma **fórmula** que funcione bem para cada contexto.[1]

---

1 PETER WAYNER. Disappearing Cryptography : Information Hiding: Steganography & Watermarking. Amsterdam: Morgan Kaufmann, 2002. v. 2nd edISBN 9781558607699.

Por que comprimir?

- Necessidade de economizar espaço em disco
  - Também permite reduzir tempo de transmissão
- Redundância presente em grande parte dos arquivos
  - Pouco conteúdo de informação
  - A redução gerada pela compressão varia dependendo do tipo de arquivo
    - Arquivos texto: de 20% a 50%
    - Arquivos binários: de 50% a 90%
- Compressão vs compactação
- Diversos exemplos de compressão

Compressão

- Dada uma mensagem M contendo n símbolos m_1, m_2,…, m_n
  - Queremos determinar uma codificação para os símbolos m_i tal que minimize o comprimento médio dos códigos
  - Uso da probabilidade de ocorrência de cada símbolo
    - Símbolos mais frequentes são codificados com menos bits
- Métodos de compressão comparados pela **taxa de compressão**

$$\frac{C(E) - C(S)}{C(E)}$$

E ⇒ entrada
S ⇒ saída
C ⇒ função de comprimento

  - Indica o quanto de redundância foi removida da entrada

Compressão RLE

- *Run-length encoding*
  - Uma run é uma sequência de caracteres idênticos
- Transmite-se a informação da sequência, e não os caracteres
  - O par (n, ch) indica n repetições do caractere ch
  - Exemplo:
    - **BBBBAAAAAAACCCAAAAADDEFTTTAAAAAAAA**
    - Codificação: **4B7A3C5A2D1E1F3T8A**
    - Taxa de compressão: (C(E) - C(S)) / C(E) = (34 - 18) / 34 ≃ 47%
- Vantajoso em imagens binárias

Compressão RLE

- *Run-length encoding*
  - Uma run é uma sequência de caracteres idênticos
- Transmite-se a informação da sequência, e não os caracteres
  - O par (n, ch) indica n repetições do caractere ch
  - Exemplo:
    - BBBBAAAAAAACCCAAAAADDD**EF**TTTAAAAAAAA
    - Codificação: 4B7A3C5A2D**1E1F**3T8A
    - Taxa de compressão: (C(E) - C(S)) / C(E) = (34 - 18) / 34 ≃ 47%
- Vantajoso em imagens binárias

Problema: em alguns casos a codificação pode não ser vantajosa

Compressão RLE

- Solução: só codificar sequências de tamanho maior que um determinado valor
  - Como saber se o que vem a seguir é uma sequência ou não?
    - Utilizar uma tripla (m, ch, n), onde m é um marcador
    - Usar um marcador que raramente ocorre no texto
      - Se ocorrer, usar uma repetição do marcador para representá-lo
      - Sequências de marcadores não são comprimidas
    - Neste caso, a compressão se torna eficiente para sequências de no mínimo 4 caracteres iguais

Compressão RLE

- Tamanho máximo da sequência é 255, de forma a representá-la com 1 byte
  - Ou 259, se contarmos a partir de 4, que é a sequência mínima

Compressão RLE

- Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA

Desnecessário comprimir aqui, embora a codificação **(#3A)** não seja maior que a sequência original

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F R

Aqui a codificação **(#1R)** aumentaria a mensagem comprimida

Compressão RLE

➢ Exemplo (usando o caractere '#' como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F R##

A ocorrência do marcador é representada, sem compressão, pela sua repetição

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F R###4V

Compressão RLE

➢ Exemplo (usando o caractere ‘#’ como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F R###4V #11T

Compressão RLE

➢ Exemplo (usando o caractere '#' como marcador)

M = DDDDDDAAAFFFFFR#VVVVTTTTTTTTTTT

#6D AAA #5F R###4V #11T

Note que o total de caracteres resultante é 18, e não 19. O valor 11 será representado pelo char de ASCII 11.

Taxa de compressão

(C(E) - C(S)) / C(E) = (31 - 18) / 31 ≃ 41%

Compressão RLE

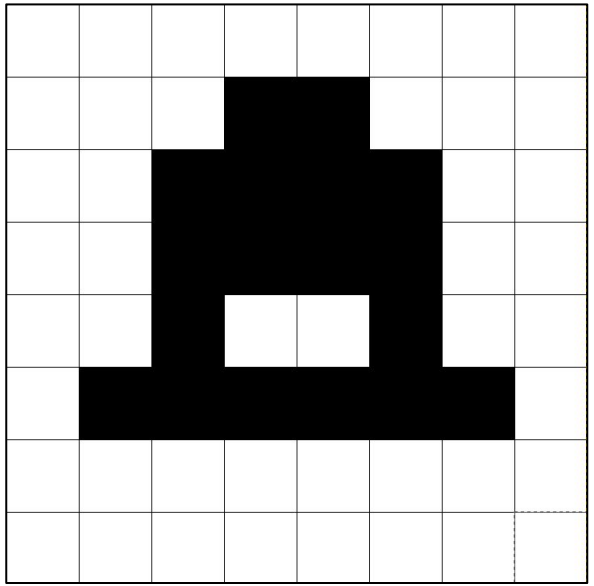
- Para mensagens binárias, não há necessidade de marcadores nem de especificar os caracteres
  - Somente contamos as repetições

Comprimida

11 2 5 4 4 4 4 1 2 1 3 6 17

Original

```
11111111
11100111
11000011
11000011
11011011
10000001
11111111
11111111
```



Taxa de compressão

(C(E) - C(S)) / C(E) = (64 - 13) / 64 ≃ 79%

Compressão de Huffman

- Proposta por David Huffman
- Algoritmo guloso que permite gerar uma codificação ótima
- Usa códigos de **comprimento variável**
- Gera uma árvore estritamente binária
  - Para n símbolos, a árvore tem n folhas e n-1 nós internos
  - É uma **árvore de prefixo**
    - Nenhum código é prefixo de outro

Compressão de Huffman

- Algoritmo
  1. Crie árvores com um único nó para cada símbolo e ordene-as pela frequência (ou probabilidade de ocorrência)
  2. Enquanto houver mais de uma árvore
     - 2.1. Selecionar t_1 e t_2 com as menores frequências
     - 2.2. Criar nova árvore em que a raiz é pai de t_1 e t_2 e possui frequência igual à soma das filhas
  3. Associar 0 com a esquerda e 1 com a direita
  4. Gerar o código do símbolo a partir do percurso da raiz até a folha onde ele se encontra

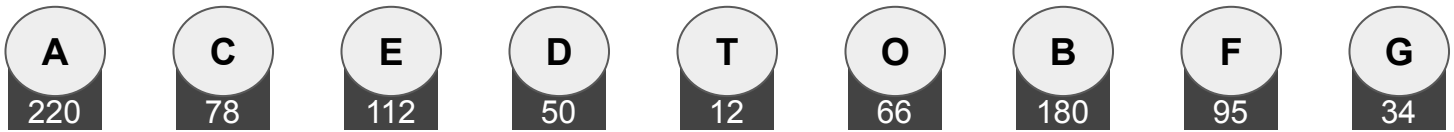
Exemplo

- Primeiro passo: determinar a frequência dos símbolos
  - Suponha os símbolos da tabela abaixo, com suas respectivas frequências de ocorrência

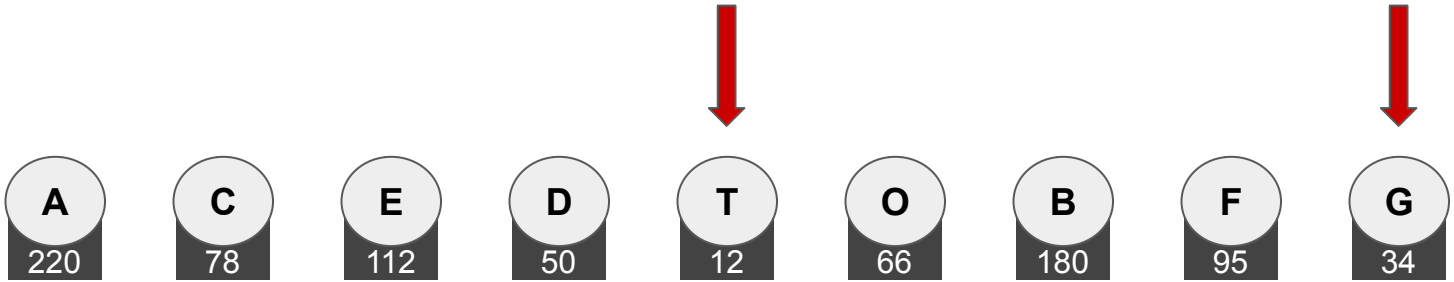
A	C	E	D	T	O	B	F	G
220	78	112	50	12	66	180	95	34

Exemplo

- Em seguida, gerar as tries unitárias

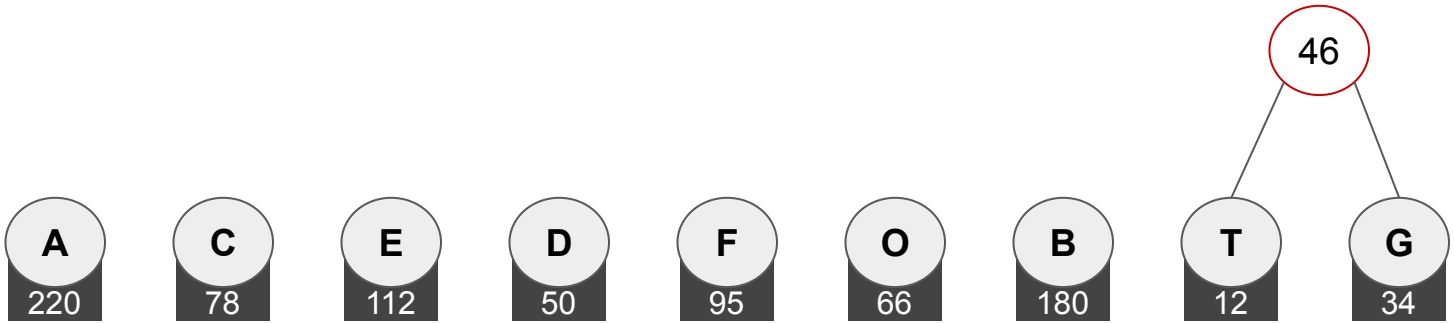


Exemplo



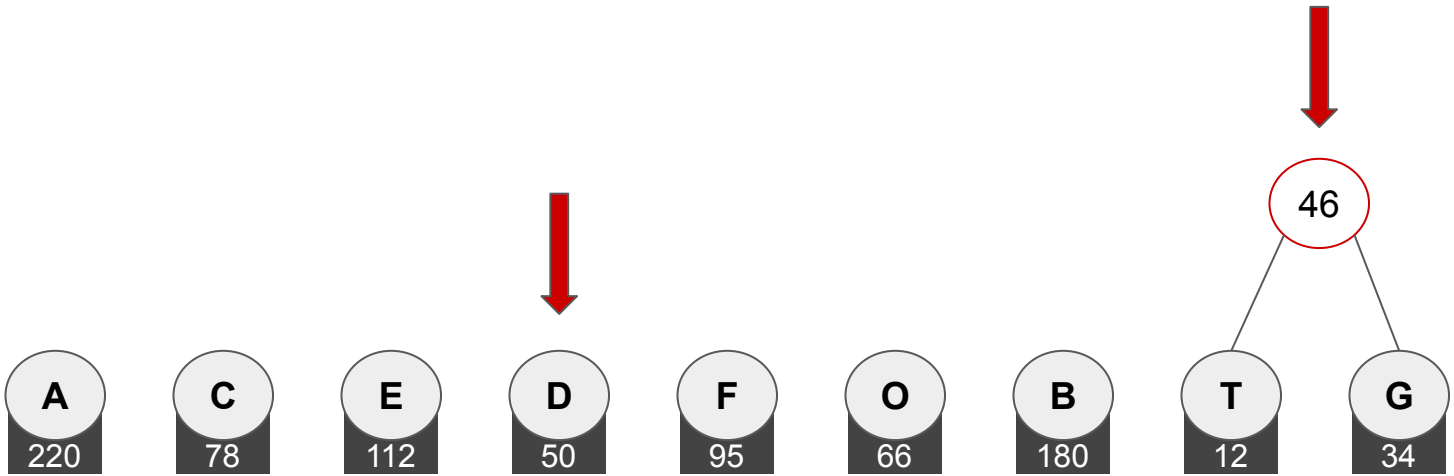
Enquanto houver mais de uma árvore, selecionar as duas de menor frequência

Exemplo

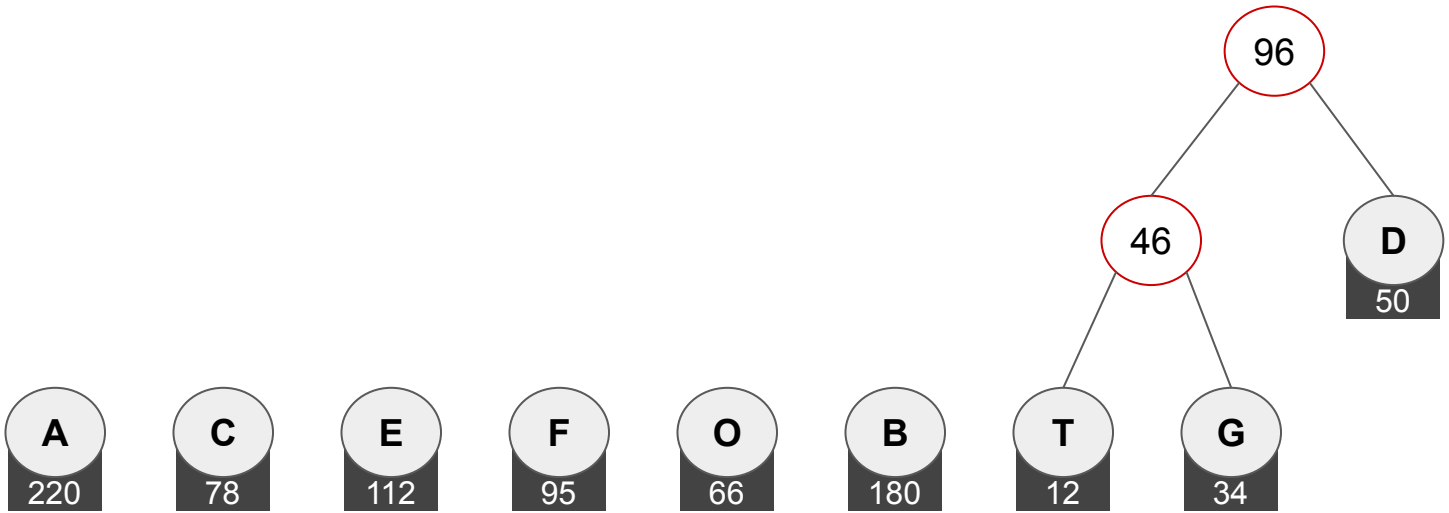


Uni-las sob uma raiz com a soma das frequências

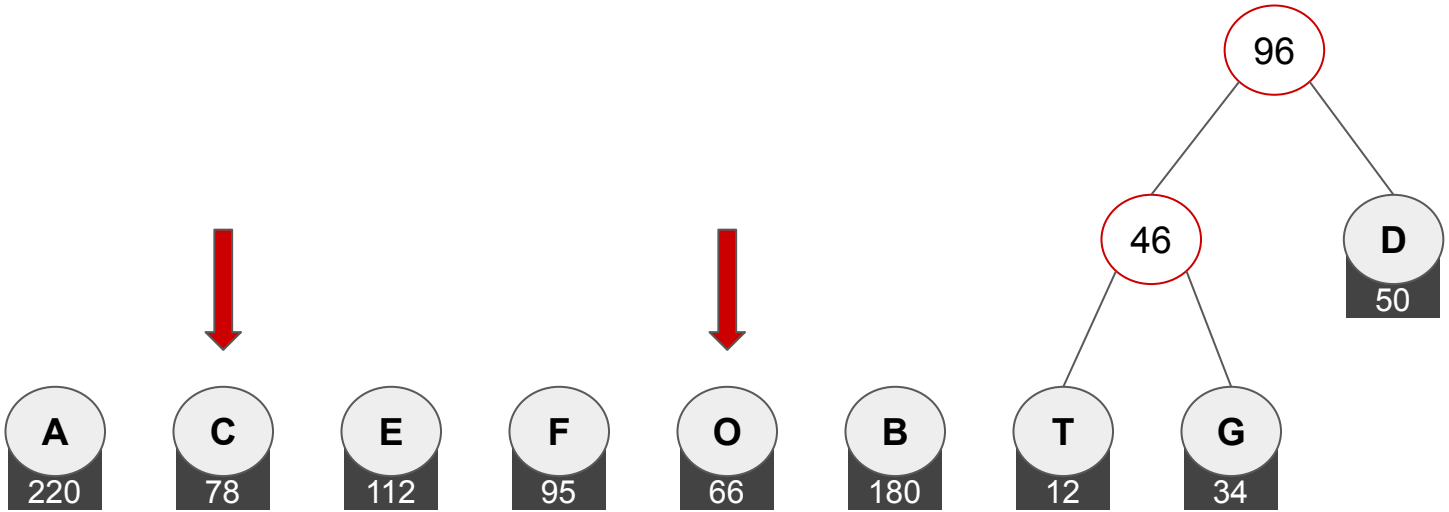
Exemplo



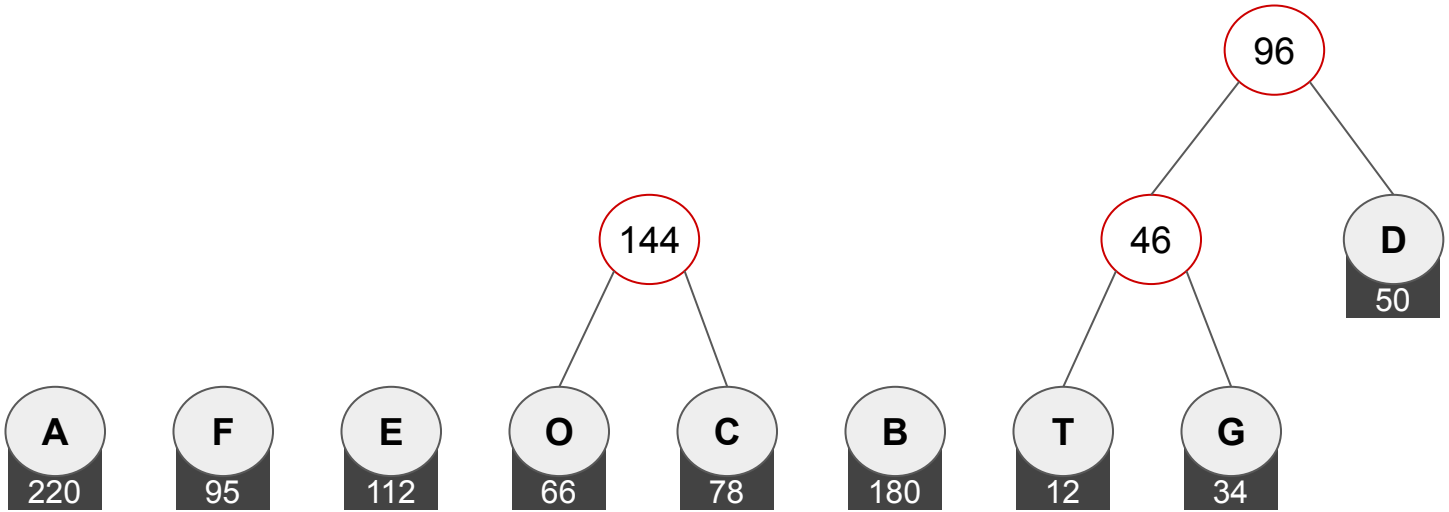
Exemplo



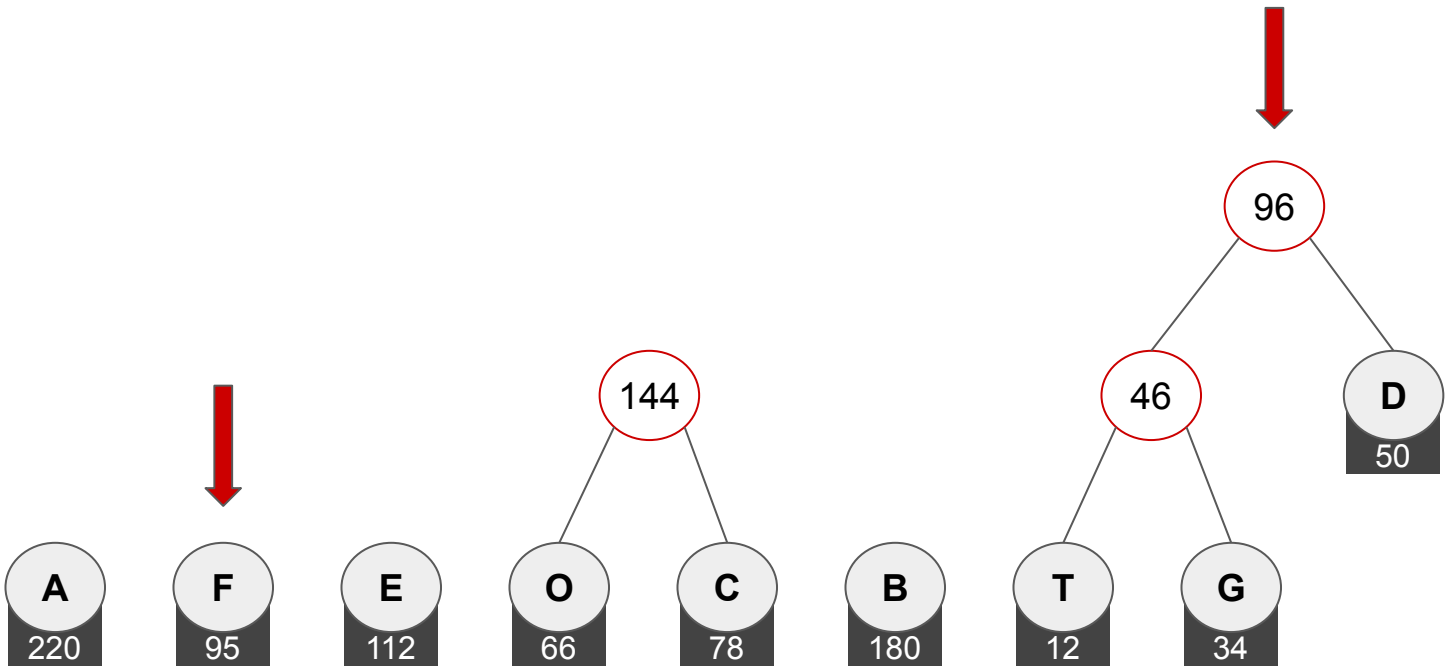
Exemplo



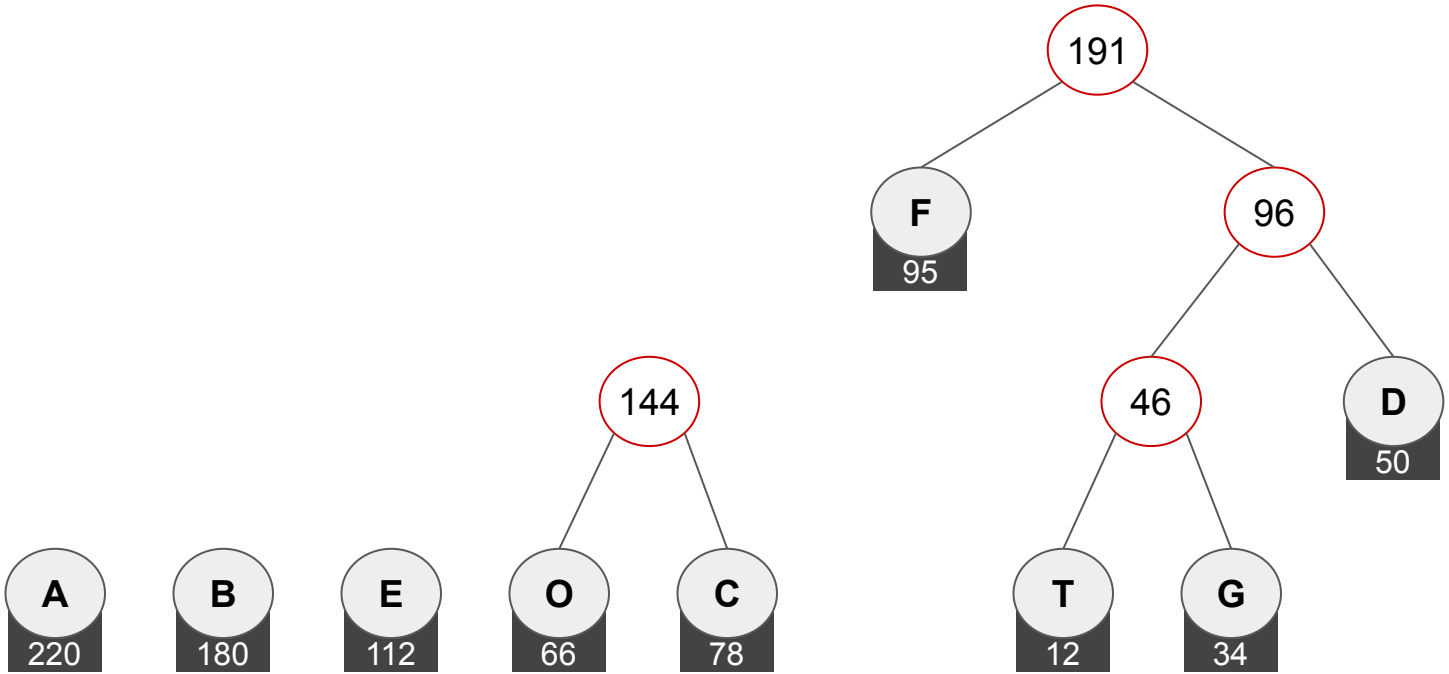
Exemplo



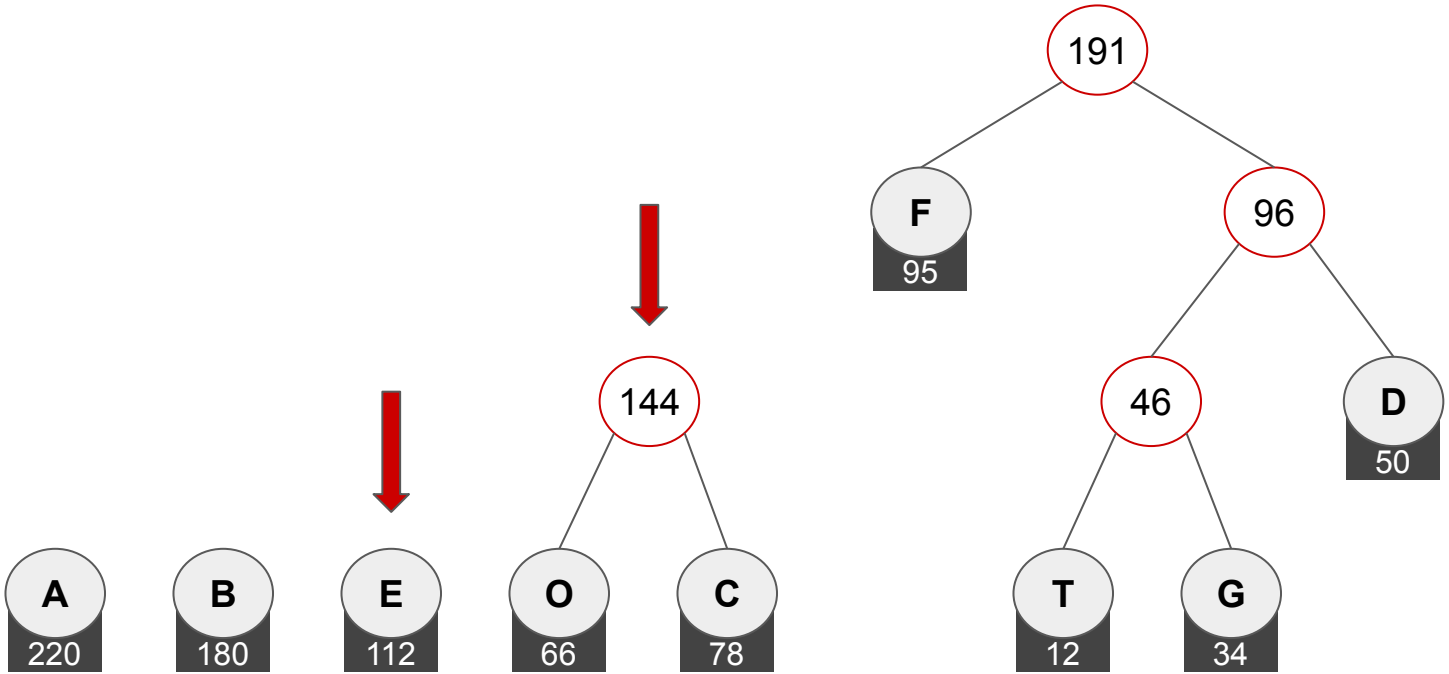
Exemplo



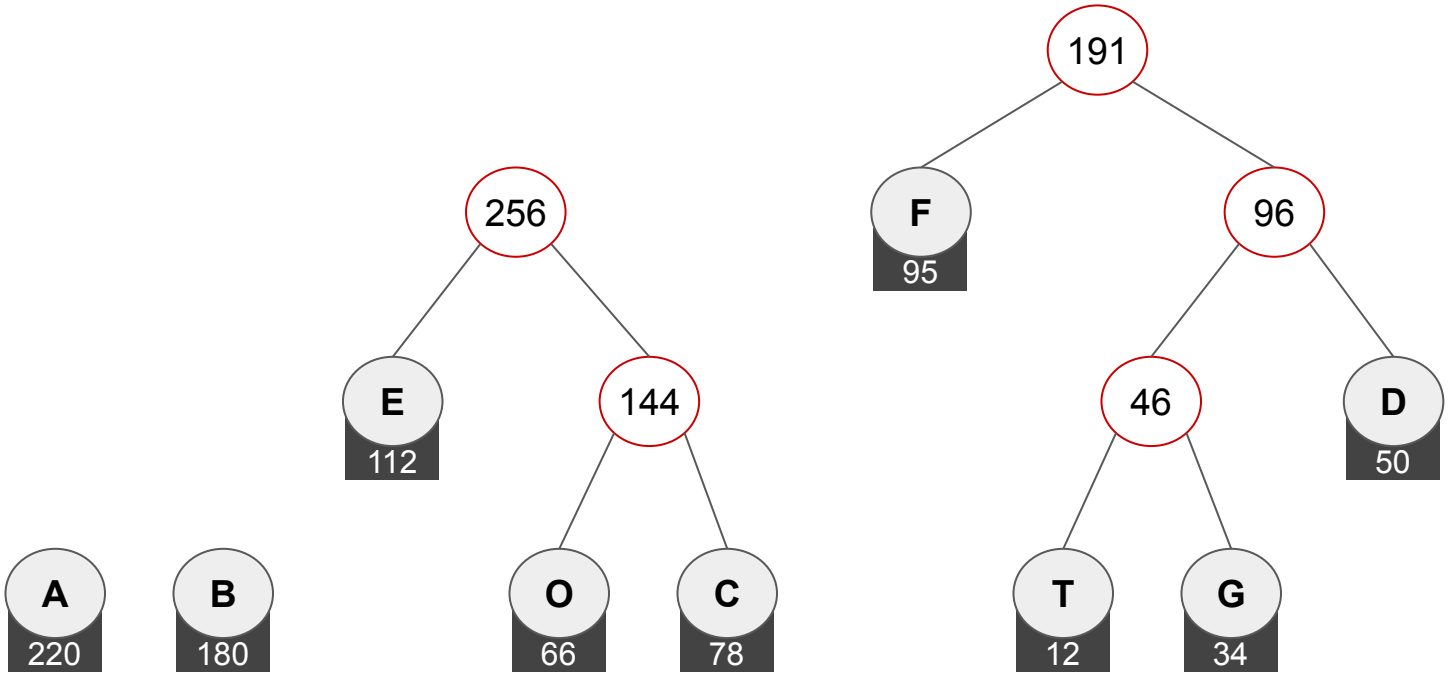
Exemplo



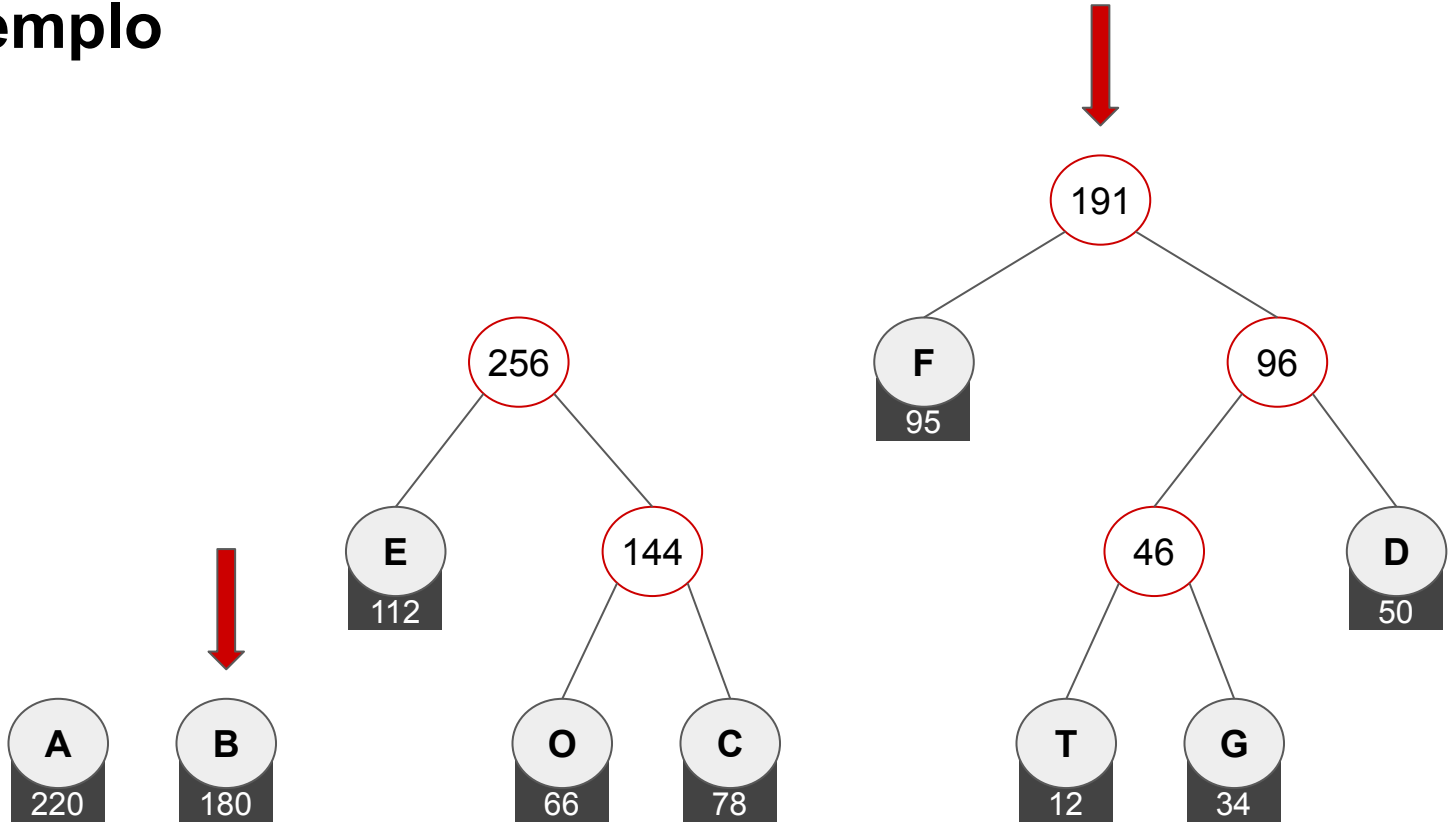
Exemplo



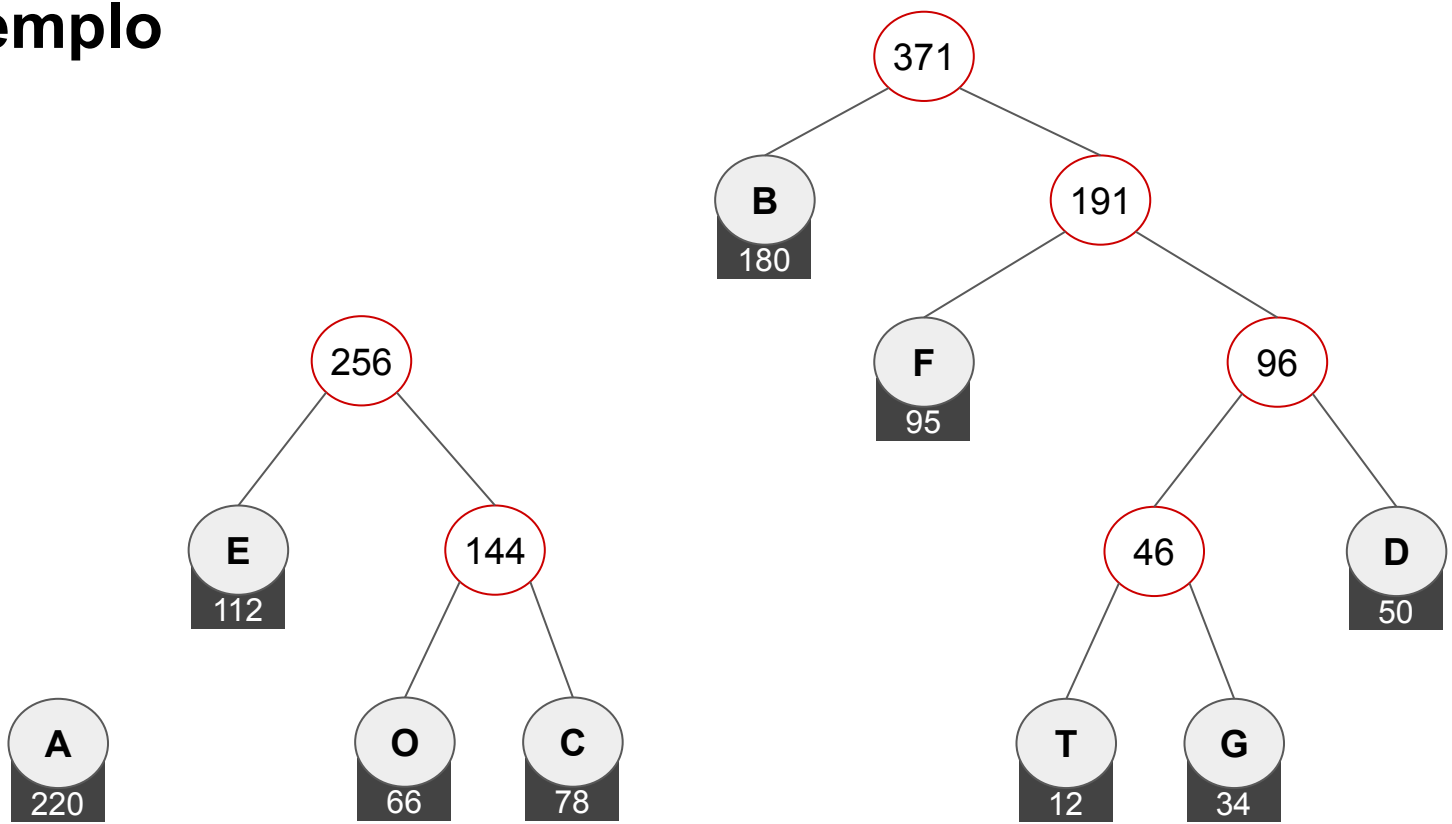
Exemplo



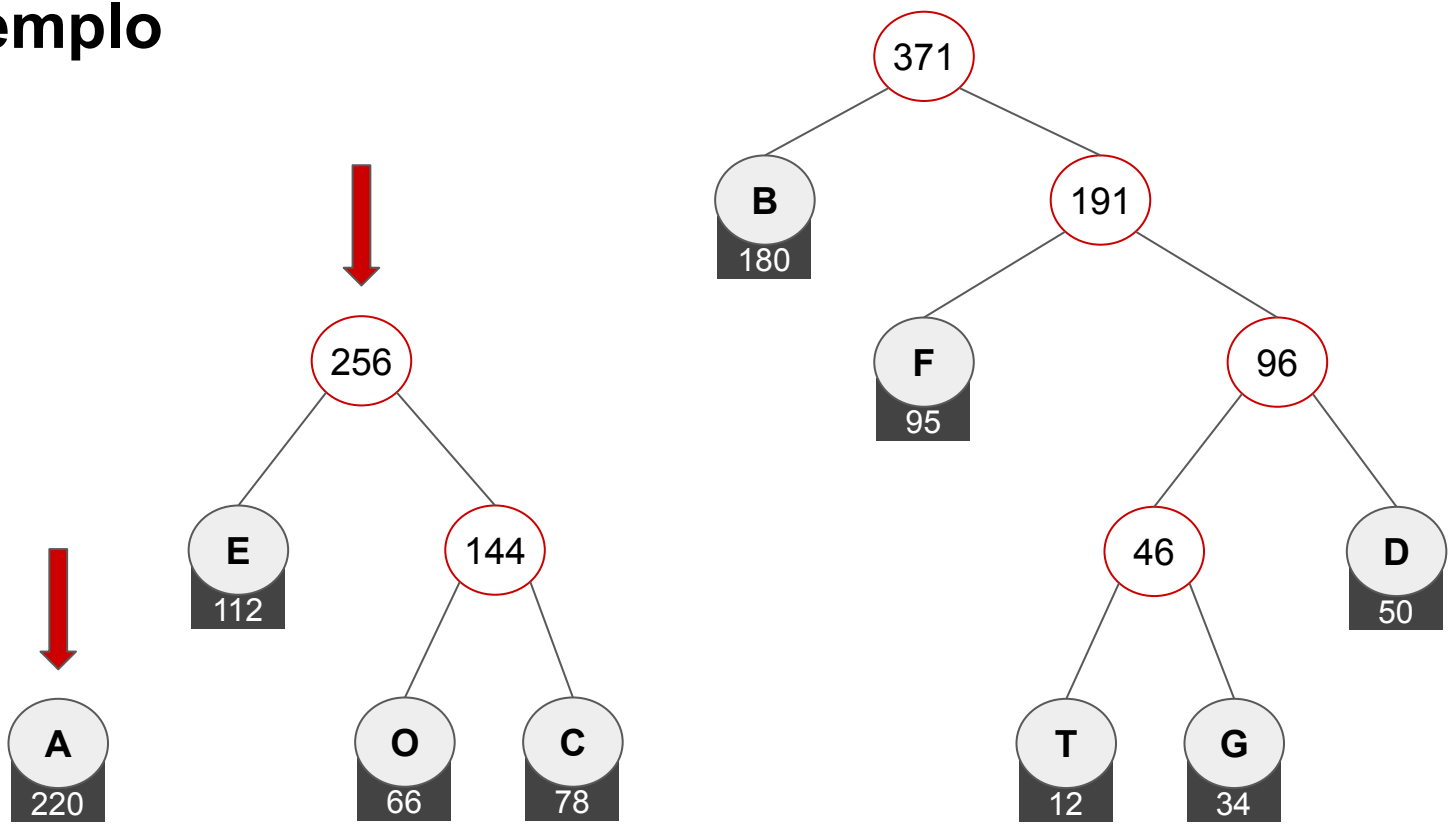
Exemplo



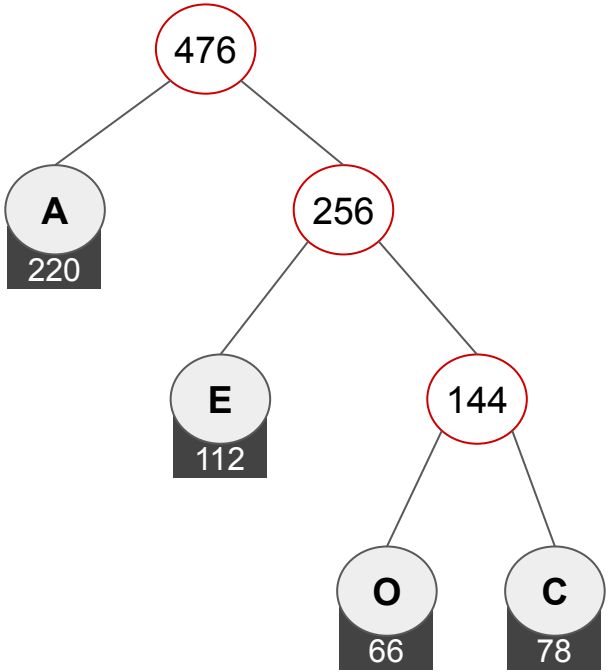
Exemplo

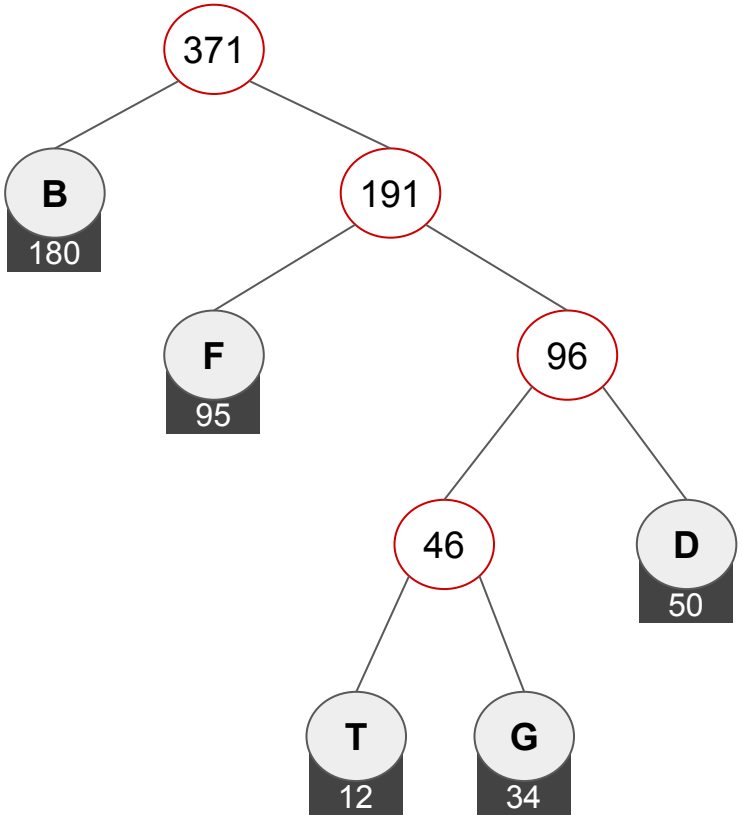


Exemplo

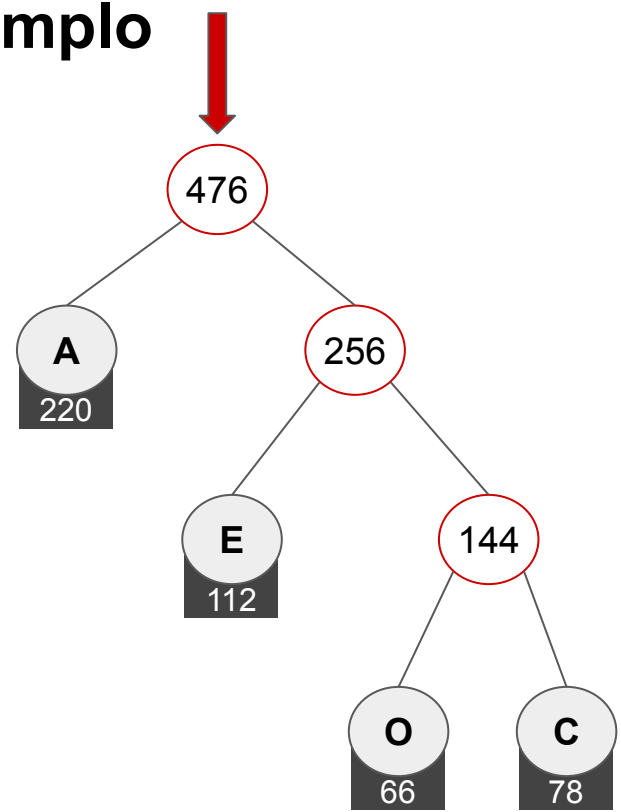


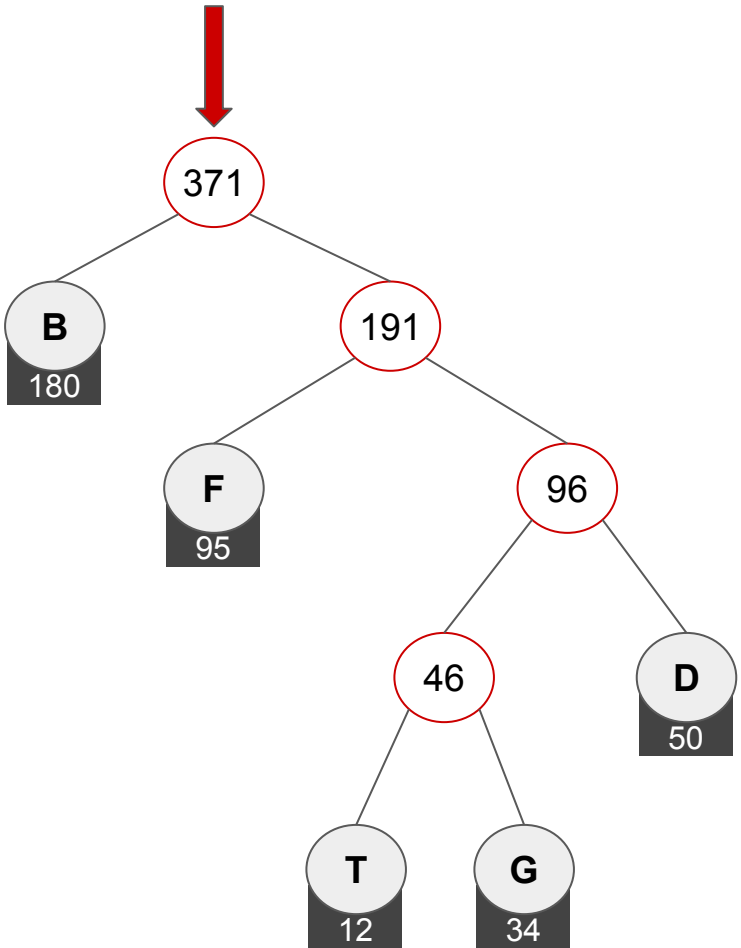
Exemplo



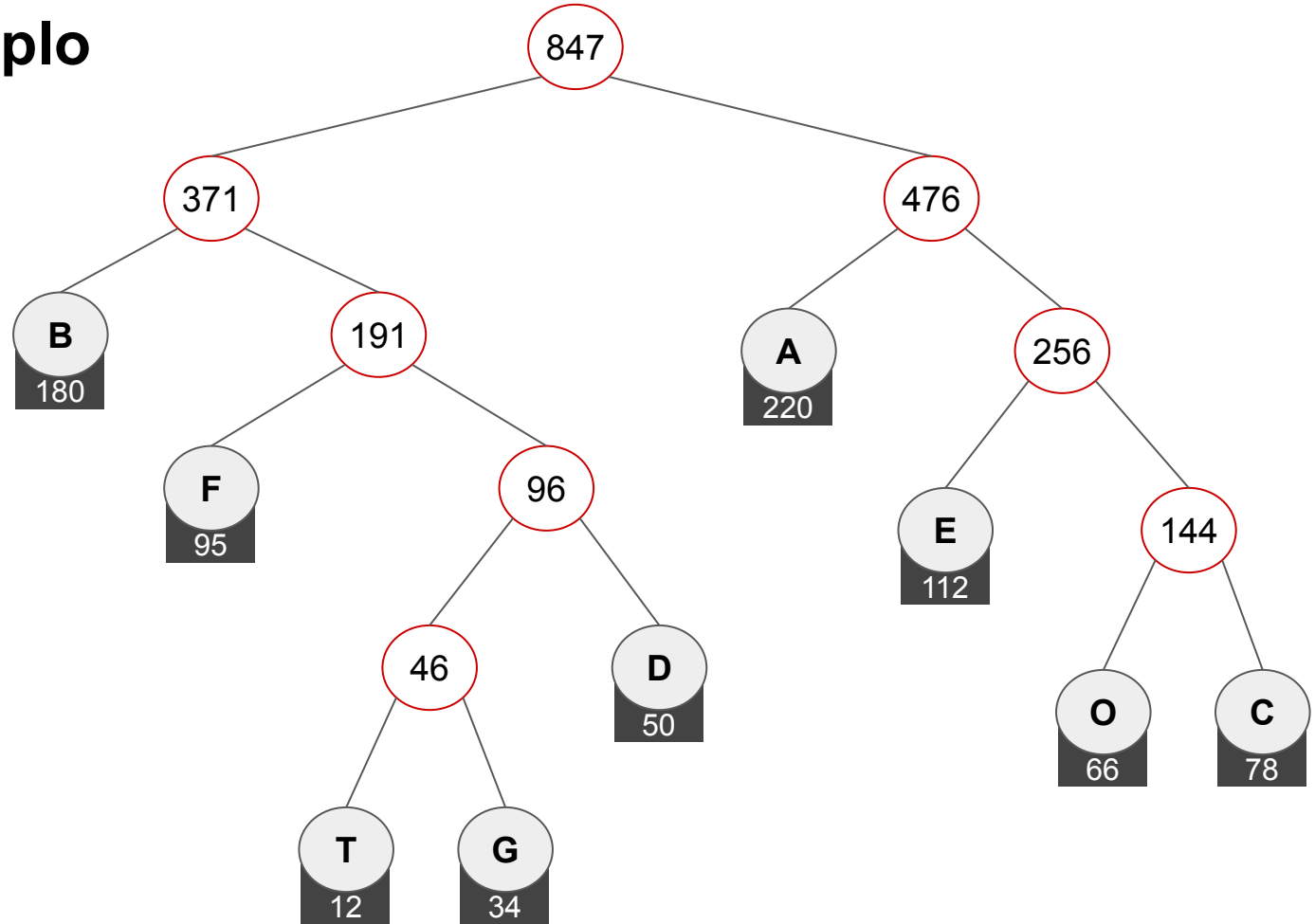


Exemplo

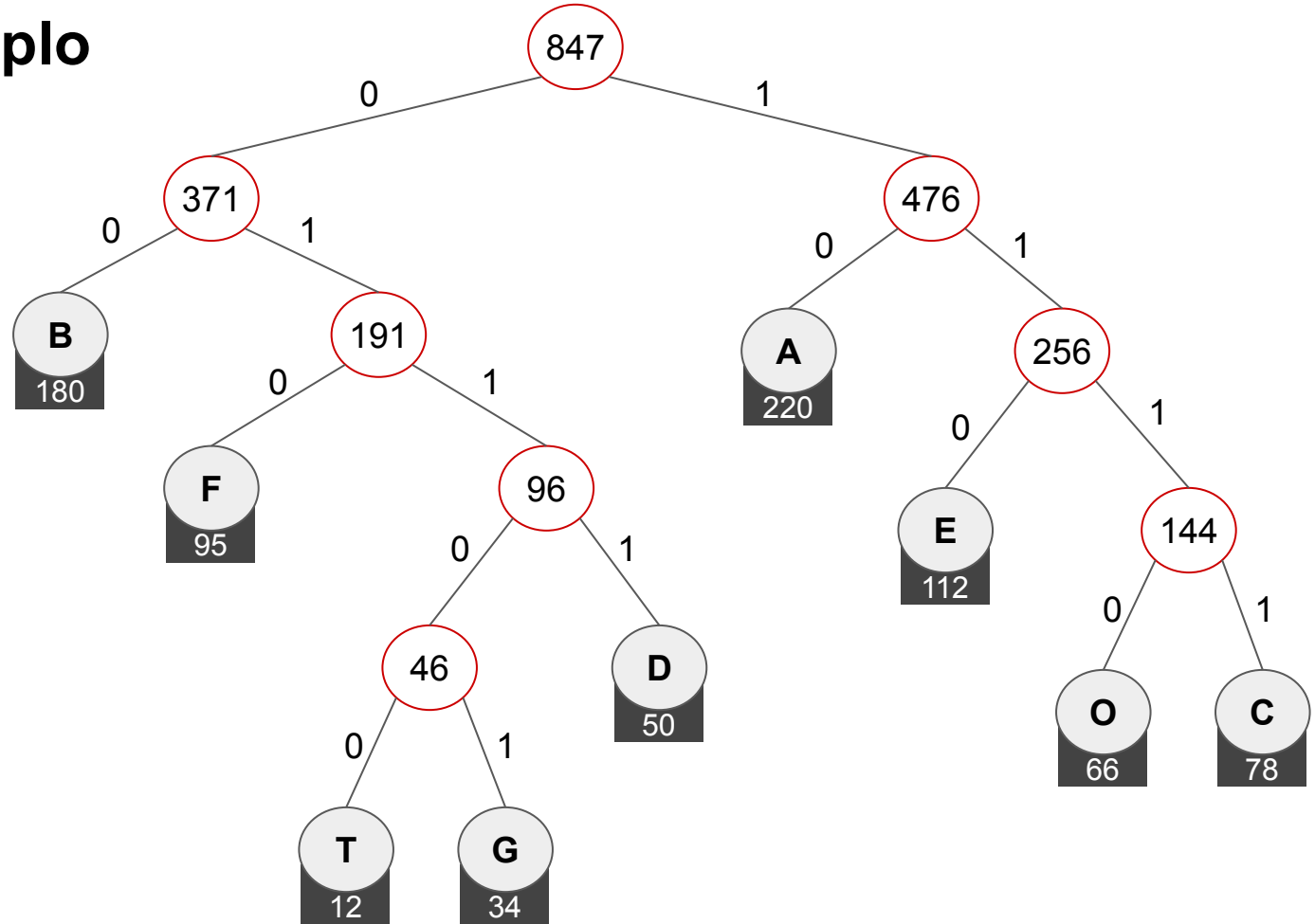




Exemplo


847
371
476
B
180
191
A
220
256
F
95
96
E
112
144
46
D
50
O
66
C
78
T
12
G
34

Exemplo


847
0
1
371
476
0
1
0
1
B
180
191
A
220
256
0
1
0
1
F
95
96
E
112
144
0
1
0
1
46
D
50
O
66
C
78
0
1
T
12
G
34

Exemplo

- A tabela final com os códigos de Huffman para cada símbolo
  - Note que os símbolos de maior frequência possuem uma codificação menor
  - Nenhum código é prefixo de outro

A	C	E	D	T	O	B	F	G
220	78	112	50	12	66	180	95	34
10	1111	110	0111	01100	1110	00	010	01101

Exemplo

- Suponha a mensagem abaixo:
  - M = ABACATEABAFABAFO
  - Compressão de Huffman
    - A B A C A T E A B A F A B A F O
    - 10 00 10 1111 10 01100 110 10 00 10 010 10 00 10 010 1110

A	C	E	D	T	O	B	F	G
220	78	112	50	12	66	180	95	34
10	1111	110	0111	01100	1110	00	010	01101

Taxa de compressão

(C(E) - C(S)) / C(E) = (16 - 42) / 16 ≃ -62%

Está correto isso?

Exemplo

- Suponha a mensagem abaixo:
  - A mensagem comprimida é uma sequência de bits
  - Obter os códigos ASCII para cada conjunto de 8 bits

A	B	A	C	A	T	E	A	B	A	F	A	B	A	F	O
10	00	10	1111	10	01100	110	10	00	10	010	10	00	10	010	1110
10001011 **139**				11100110 **230**		01101000 **104**			10010100 **148**			01001011 **75**			10 **2**

Taxa de compressão

(C(E) - C(S)) / C(E) = (16 - 6) / 16 ≃ 63%

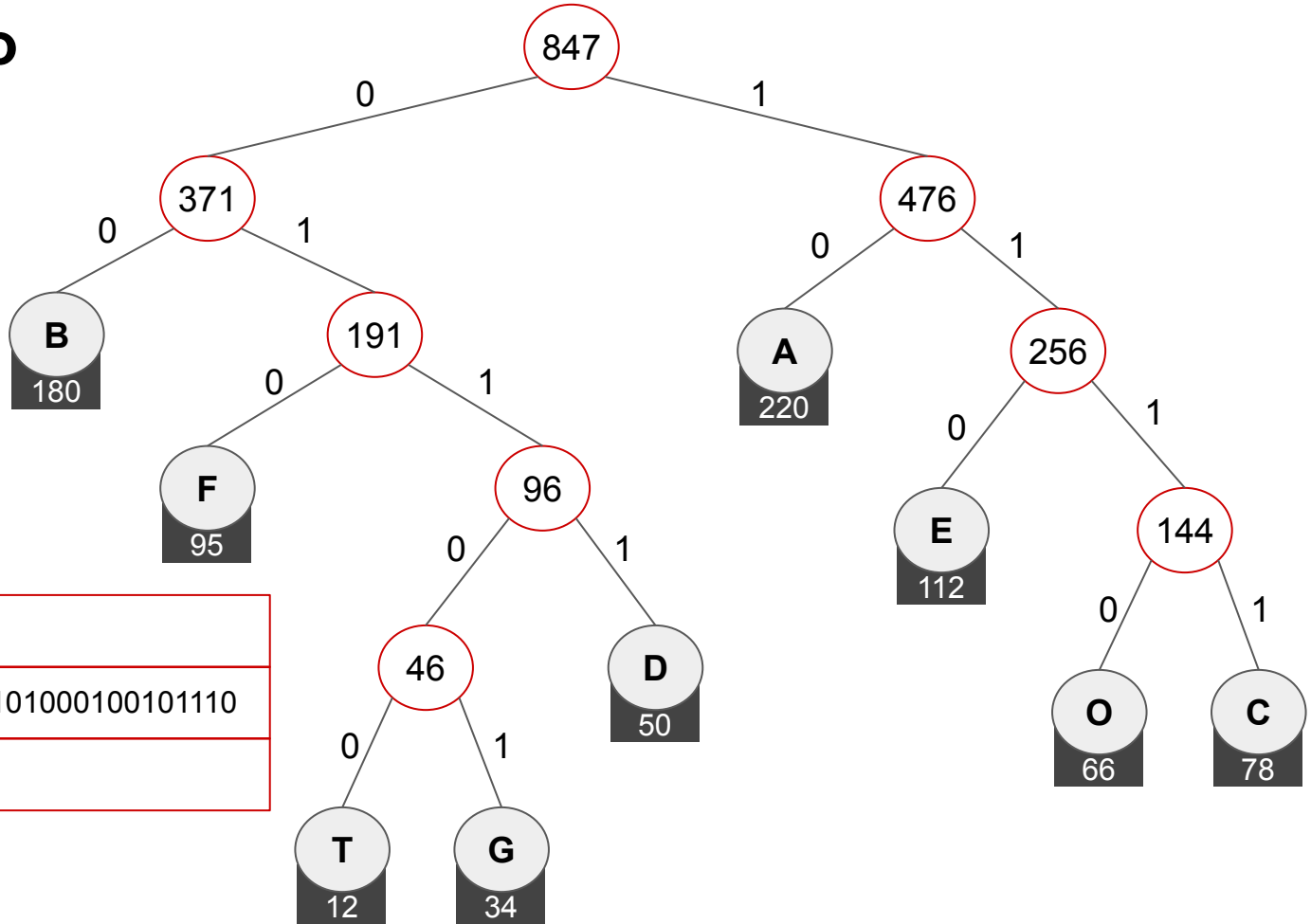
Implementação

- Várias formas de se implementar o algoritmo de Huffman
  - Uma forma natural é o uso de uma min-heap
  - Algoritmo executado em O(n log n)
    - n é o número de símbolos
    - n-1 uniões de tries, com O(log n) para cada operação na heap
  - O cômputo da tabela de frequências pode ser realizado em O(m), onde m é o tamanho da mensagem M

Decodificação

- O processo de decodificação é simples
  - Basta percorrer iterativamente a árvore da raiz até as folhas, seguindo o caminho especificado pela sequência de bits da mensagem codificada
  - A decodificação termina quando se atinge o fim da mensagem

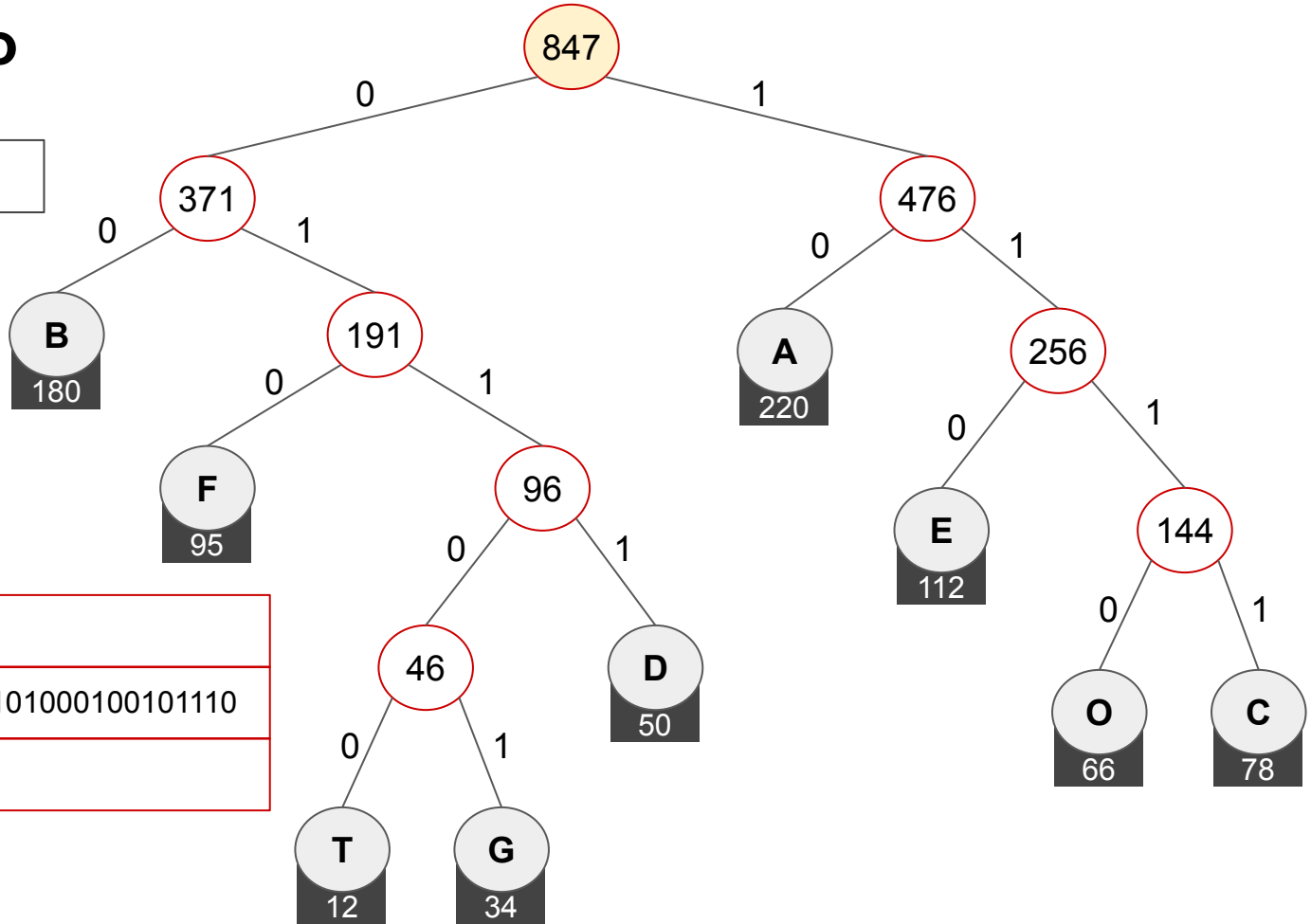
Decodificação



EXEMPLO
1000101111100110011010001001010001001011 10

Decodificação

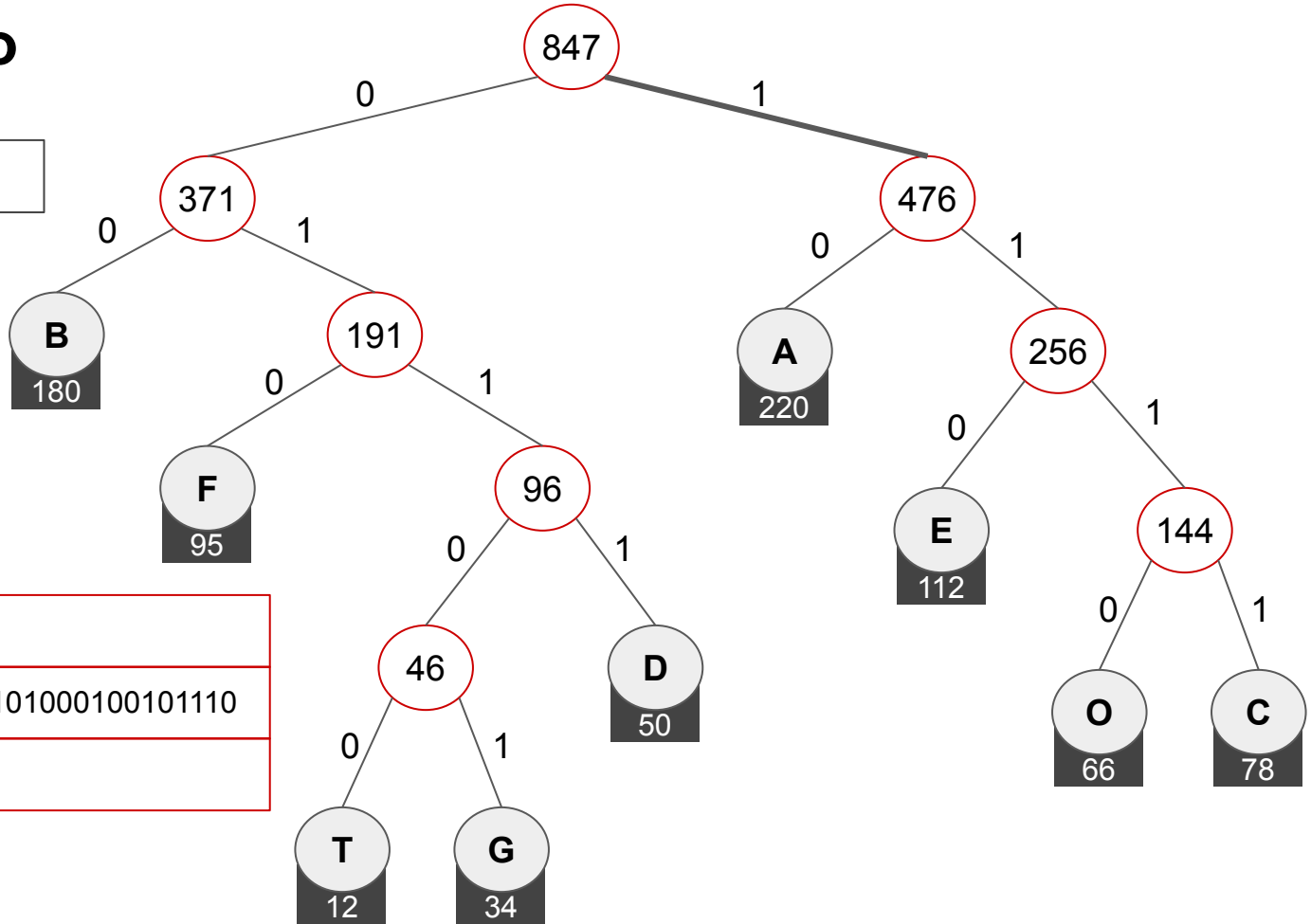
Começa na raiz...



EXEMPLO
1000101111100110011010001001010001001011 10

Decodificação

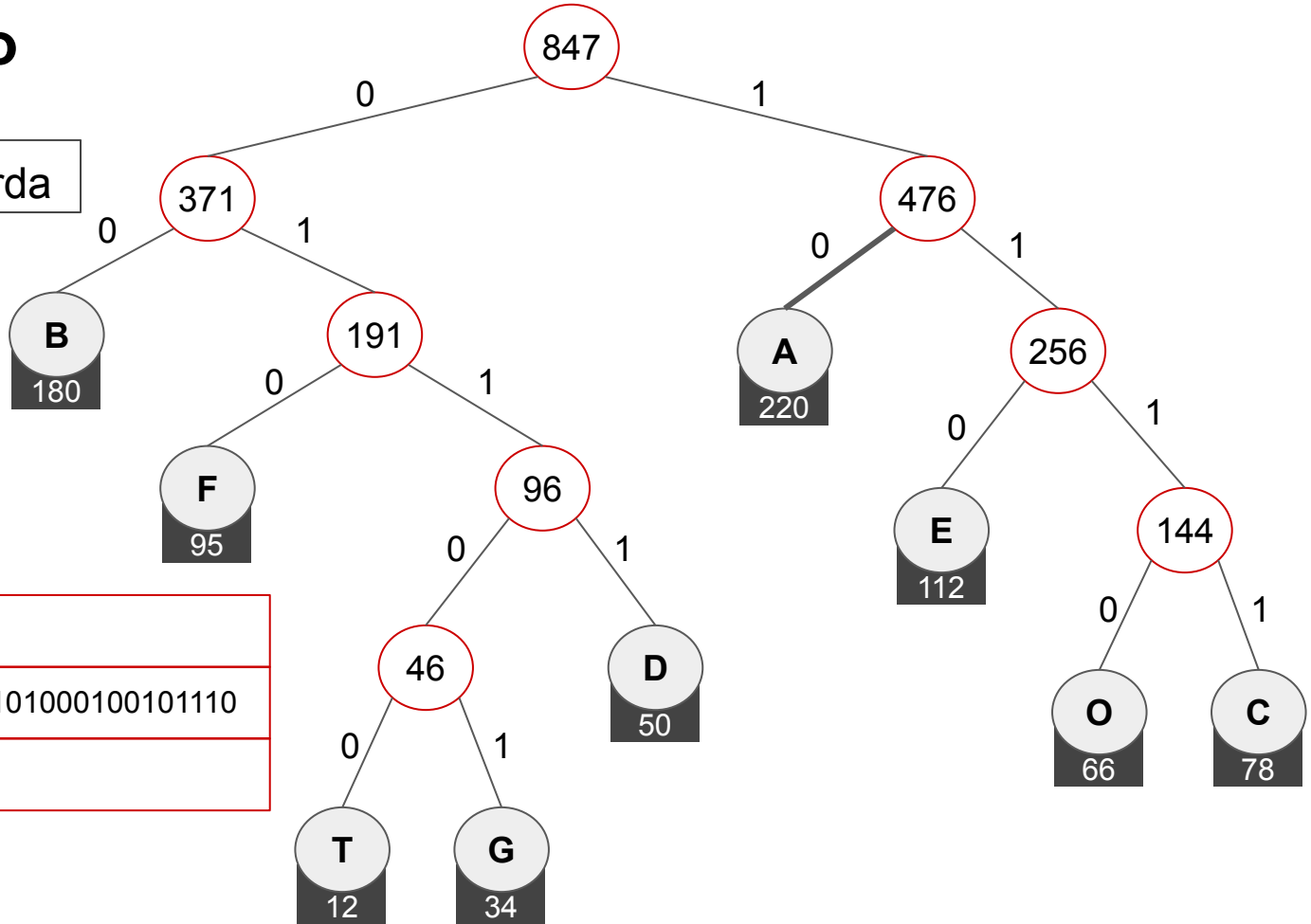
Bit 1 ⇒ direita



EXEMPLO
1000101111100110011010001001010001001011 10

Decodificação

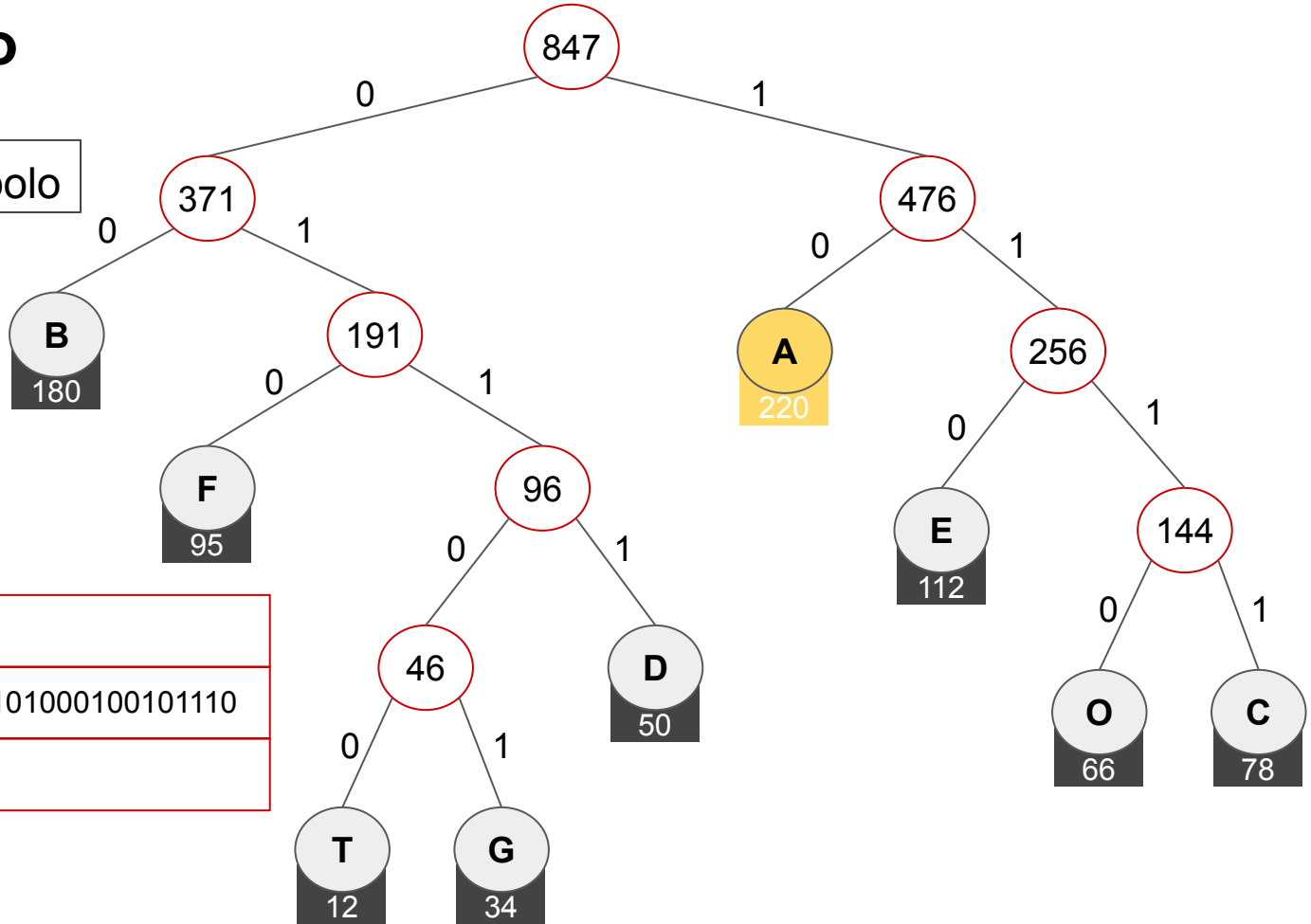
Próximo bit: 0 ⇒ esquerda



EXEMPLO
100010111110011001101000100101000100101110

Decodificação

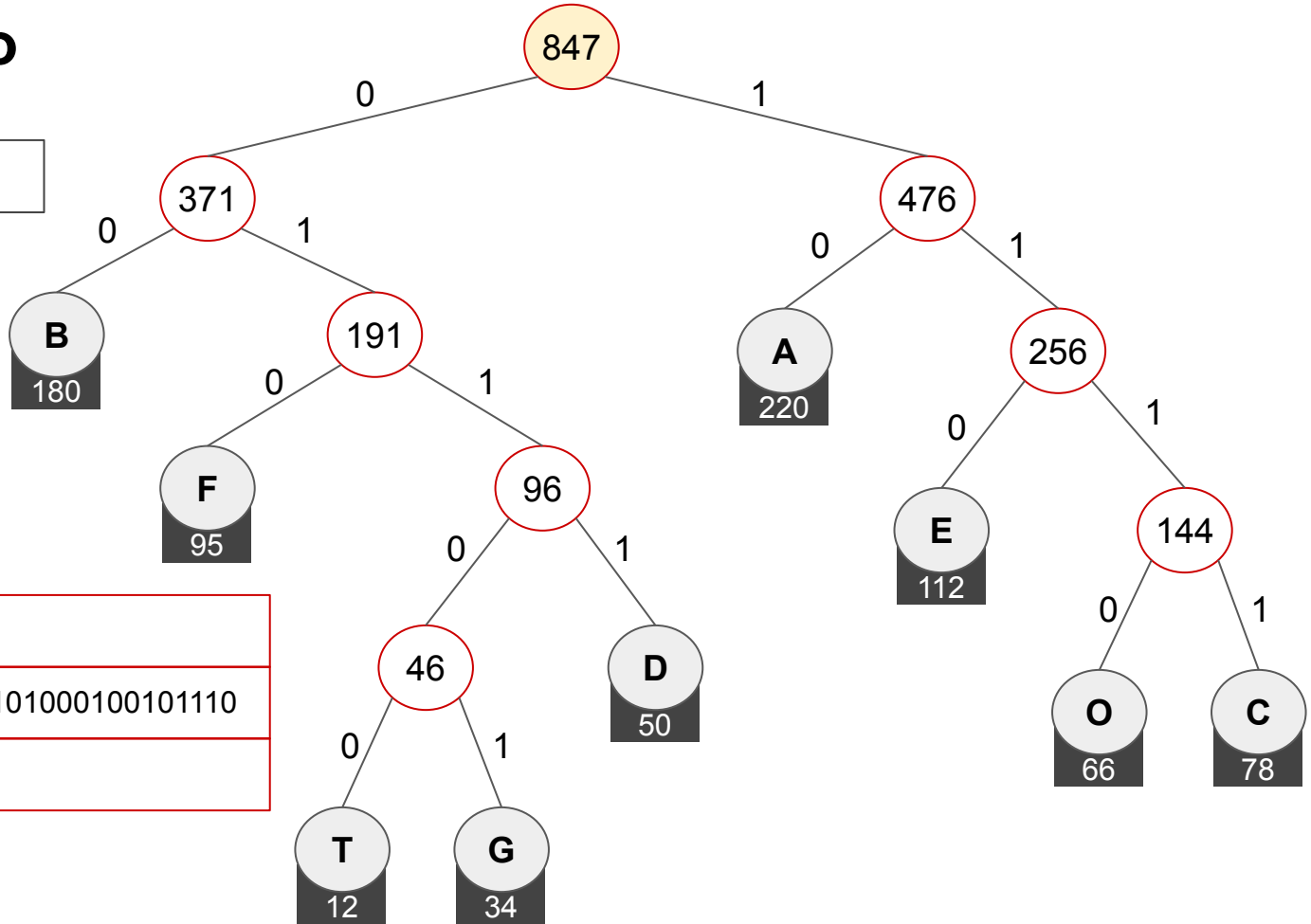
Atingiu folha: emite símbolo



EXEMPLO
10001011111001100110100010010100010010111 0
A

Decodificação

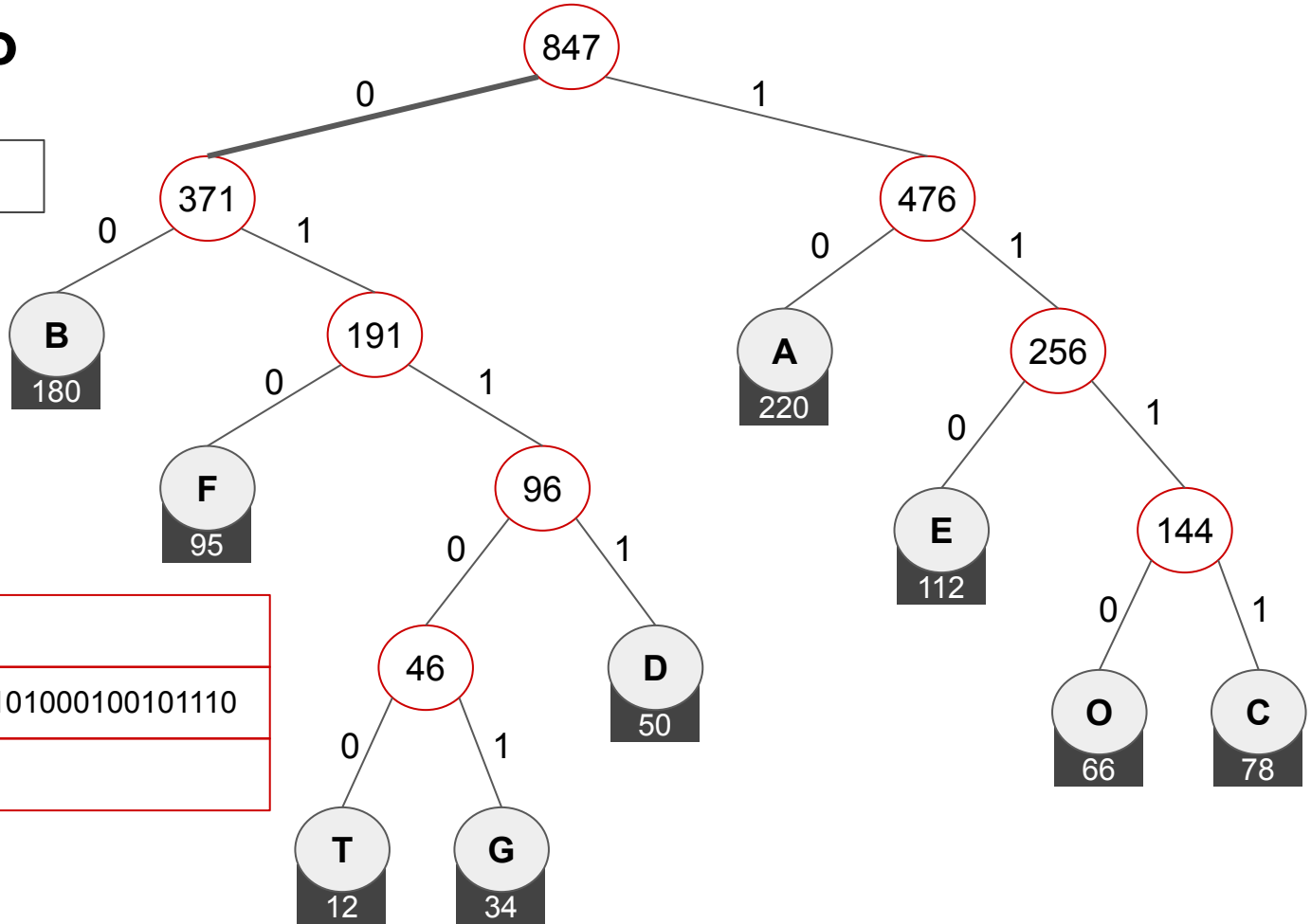
Volta na raiz



EXEMPLO
10001011111001100110100010010100010010111 0
A

Decodificação

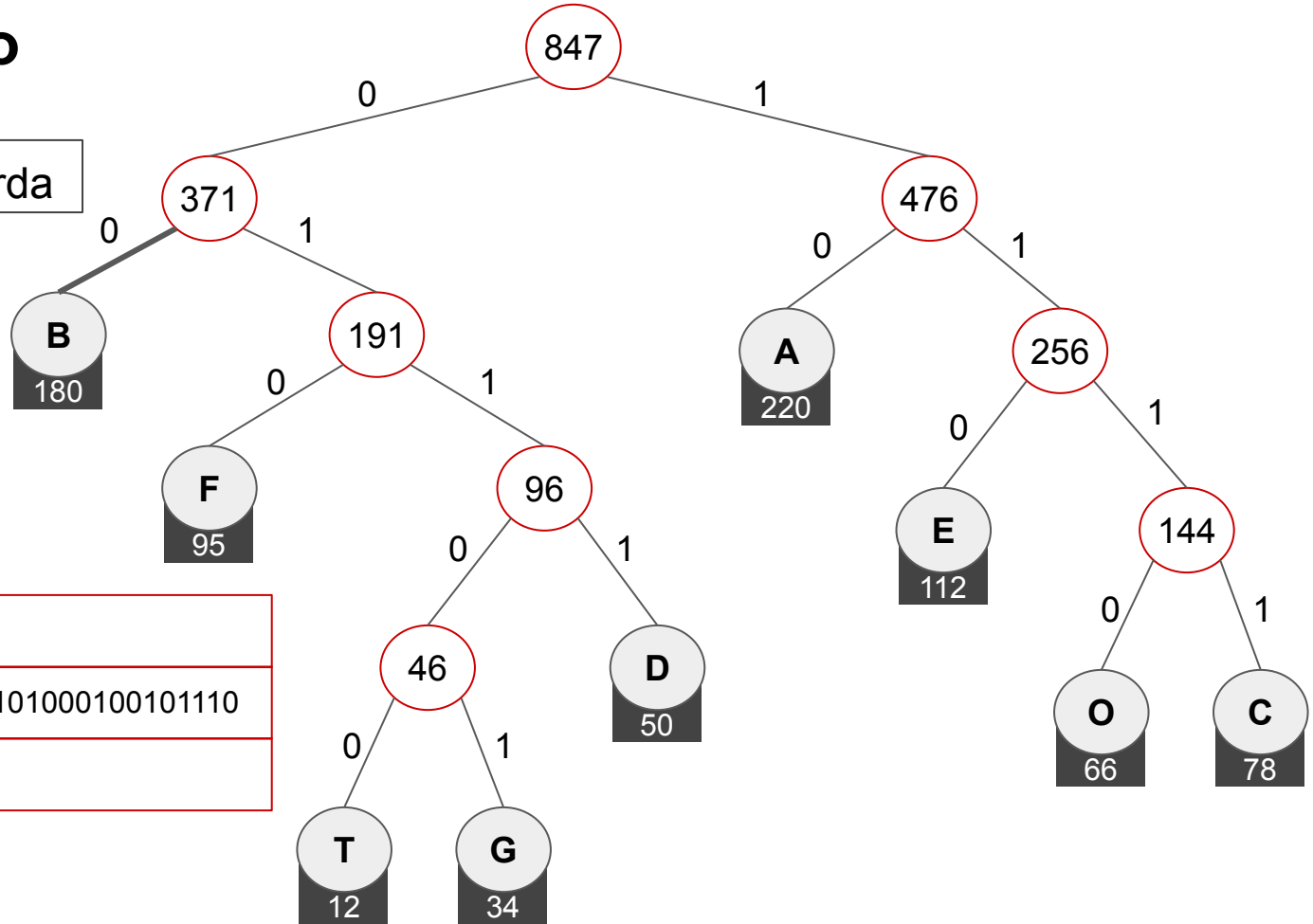
Bit 0 ⇒ esquerda



EXEMPLO
1000101111100110011010001001010001001011 10
A

Decodificação

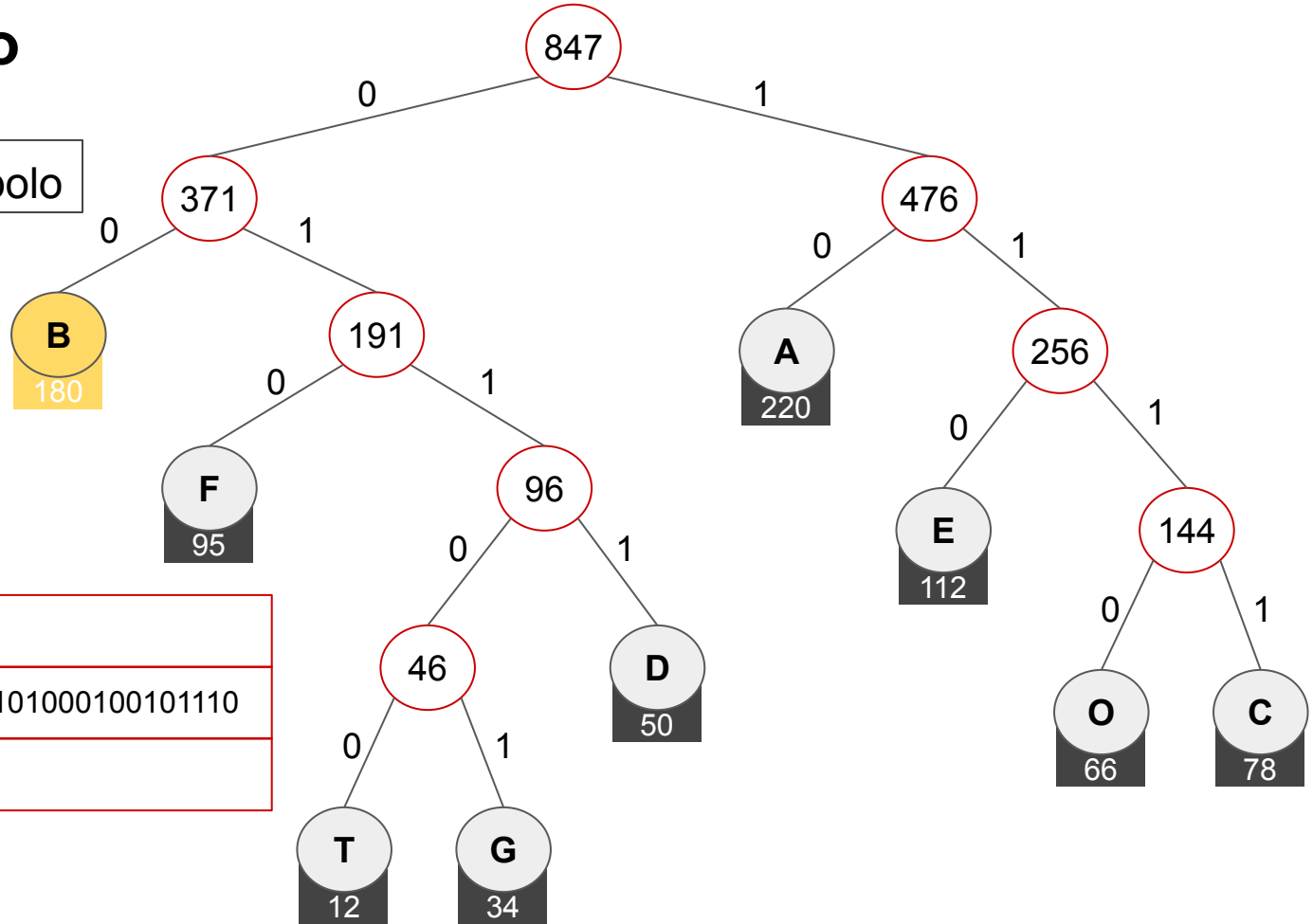
Próximo bit: 0 ⇒ esquerda



EXEMPLO
1000101111100110011010001001010001001011 10
A

Decodificação

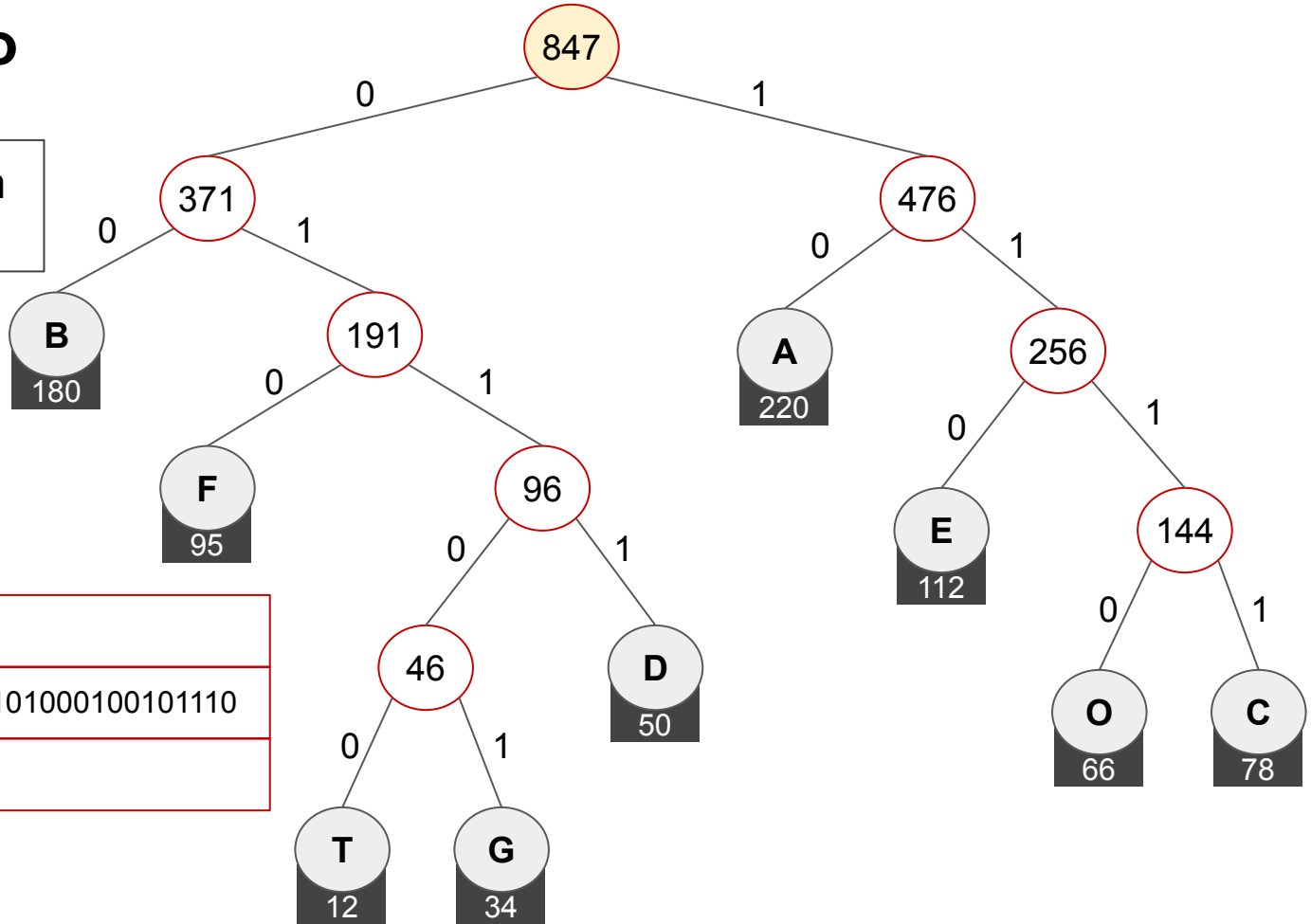
Atingiu folha: emite símbolo



EXEMPLO
1000101111100110011010001001010001001011 10
AB

Decodificação

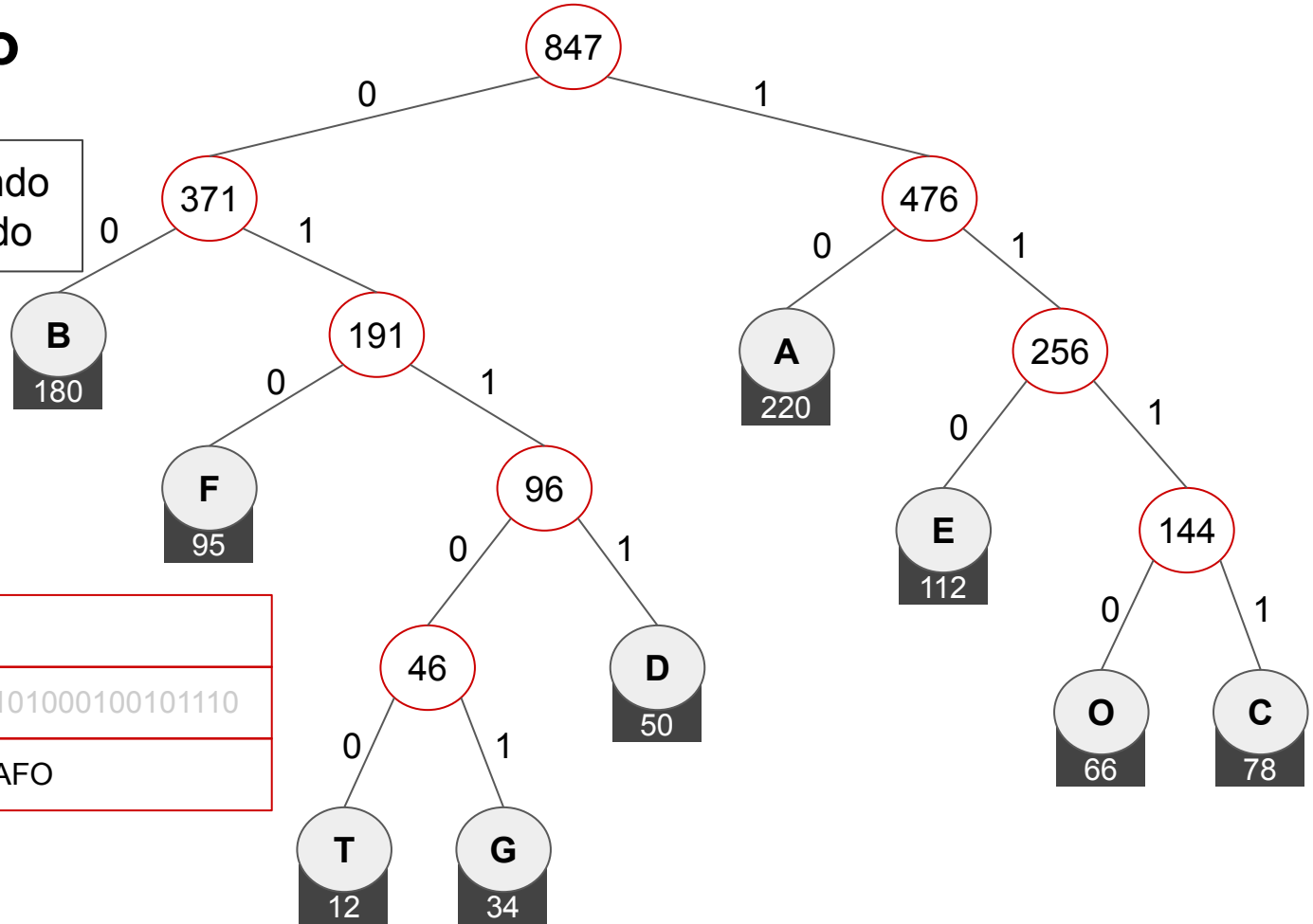
Volta na raiz… (e assim por diante)



EXEMPLO
1000101111100110011010001001010001001011 10
AB

Decodificação

O algoritmo termina quando o último bit é decodificado



EXEMPLO
1000101111100110011010001001010001001011 10
ABACATEABAFABAFO

Exercício 1

Use Huffman para codificar o texto: **ABRACADABRA**

Compressão de Huffman

- Problemas
  - A decodificação necessita da tabela de conversão
    - Pode haver um acordo prévio com relação à árvore usada
    - A tabela pode ser enviada junto com a mensagem
      - Só útil para mensagens longas ou para várias mensagens
  - Podemos não conhecer as frequências dos símbolos previamente
  - Podemos querer incluir novos símbolos na árvore

Huffman adaptativo

- Proposto por Robert G. Gallager e melhorado por Donald Knuth
- Também conhecido como algoritmo FGK
- Baseia-se na **propriedade de irmandade**
  - Se um nó (exceto a raiz) possui um irmão e o percurso em largura da direita para a esquerda gera uma lista de nós com contadores de frequência não-crescentes, pode-se provar que essa árvore é uma árvore de Huffman
- Cada nó tem um contador de frequência, que é incrementado quando há uma nova ocorrência
- Se a propriedade de irmandade é ferida, a árvore é reestruturada

Huffman adaptativo

- Como verificar a propriedade?
  - Uma lista duplamente encadeada é mantida, contendo os nós da árvore ordenados pelo percurso em largura da direita para a esquerda
  - Um **bloco**$_i$ é uma seção da lista em que cada nó tem frequência i
    - O primeiro nó do bloco é chamado de **líder** do bloco
- Símbolos não utilizados são mantidos em um único nó de frequência 0
  - Ao surgir um novo símbolo, ele é removido do nó 0 e incluído em um novo nó de frequência 1
    - Ele se torna irmão do nó 0 e filho de um outro nó de frequência também 1
- Se o símbolo já existe, incrementa sua frequência e atualiza nós

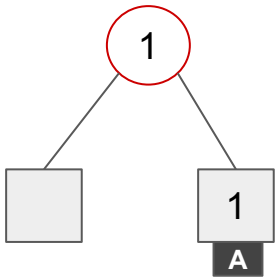
Exemplo

M = ABACATEABAFABAFO

Inicialmente a árvore só possui o nó de frequência 0

Exemplo

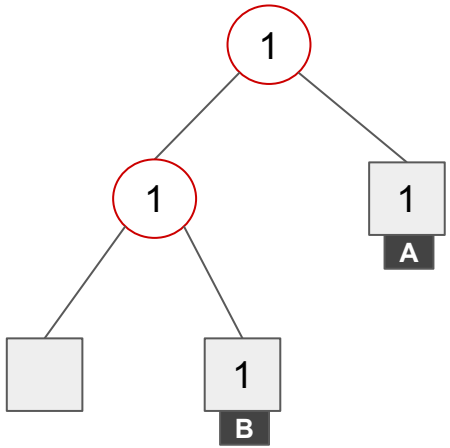
M = **A**BACATEABAFABAFO



Primeira ocorrência do caractere A

Exemplo

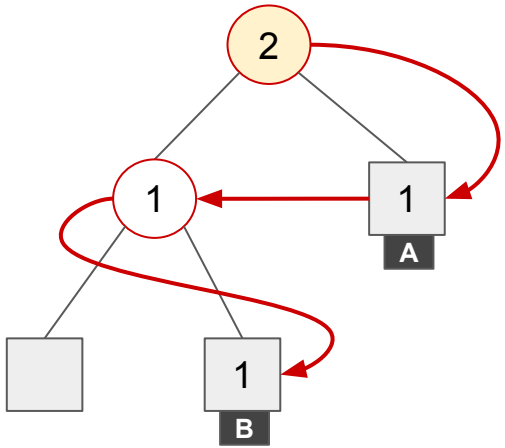
M = ABACATEABAFABAFO



Primeira ocorrência do caractere B

Exemplo

M = ABACATEABAFABAFO

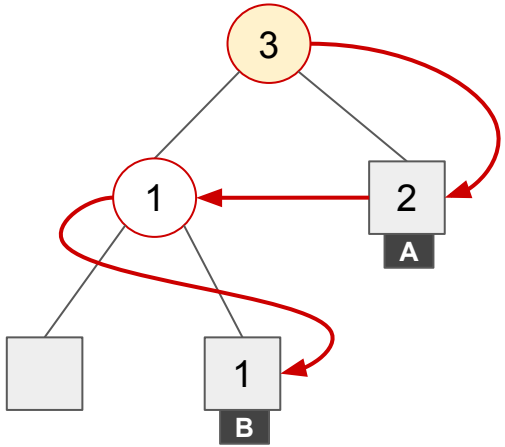


Propriedade de irmandade OK!

É preciso atualizar a frequência da raiz

Exemplo

M = ABACATEABAFABAFO

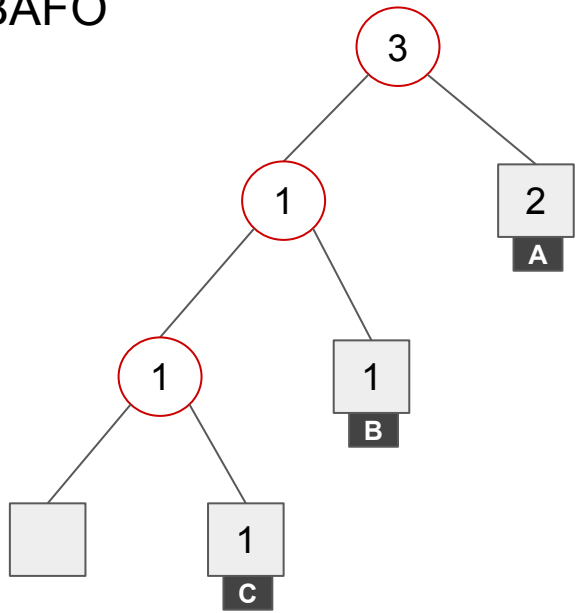


Propriedade de irmandade OK!

Segunda ocorrência do caractere A, incrementa e atualiza frequências

Exemplo

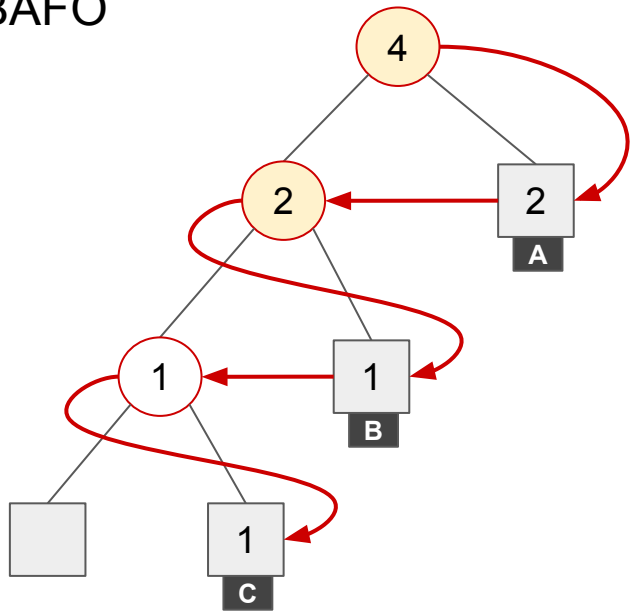
M = ABA**C**ATEABAFABAFO



Primeira ocorrência do caractere C

Exemplo

M = ABA**C**ATEABAFABAFO

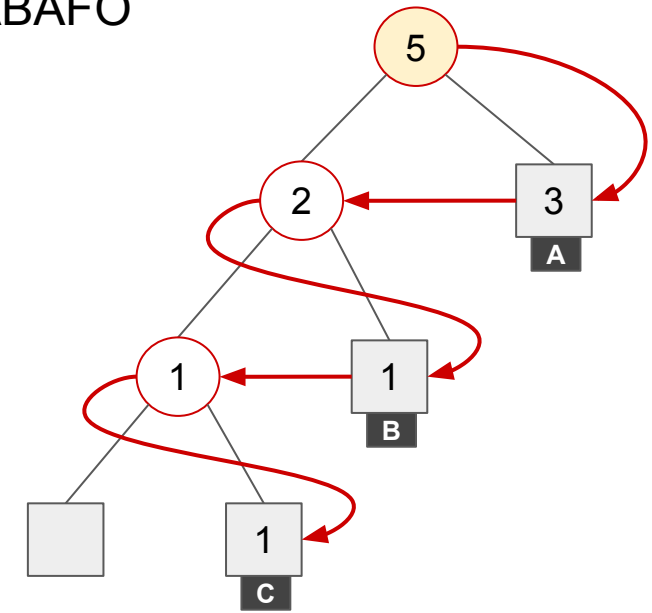


Propriedade de irmandade OK!

Atualiza nós

Exemplo

M = ABAC**A**TEABAFABAFO

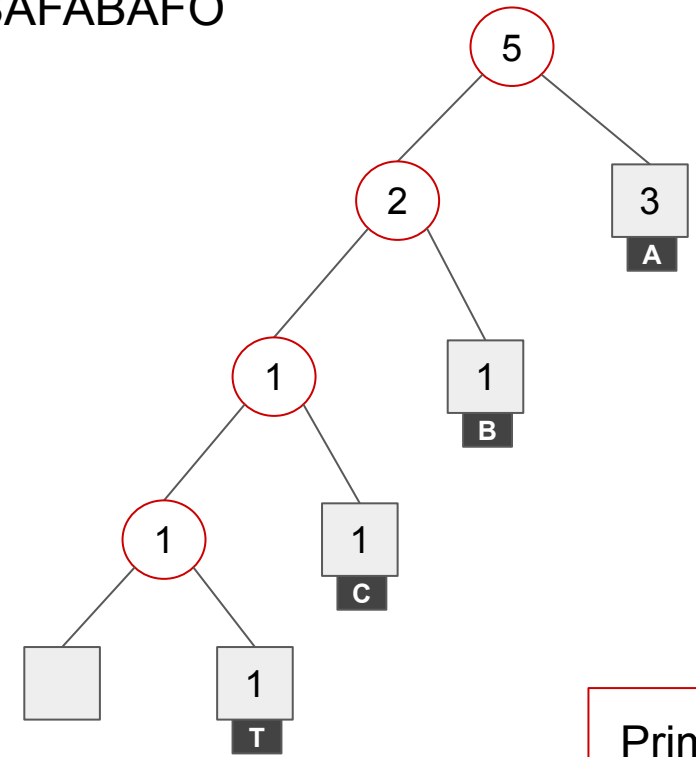


Propriedade de irmandade OK!

Terceira ocorrência de A, incrementa e atualiza frequências

Exemplo

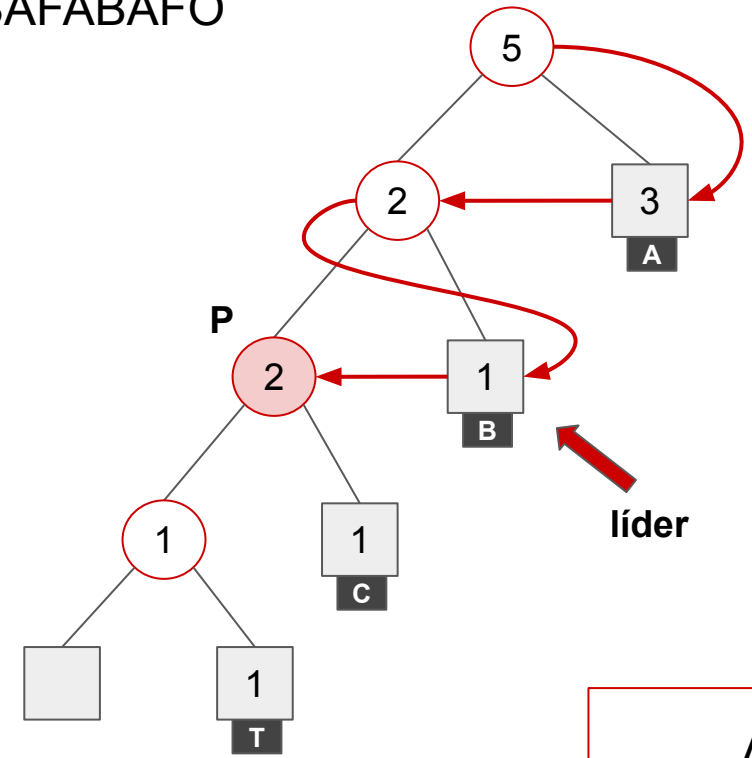
M = ABACA**T**EABAFABAFO



Primeira ocorrência de T

Exemplo

M = ABACA**T**EABAFABAFO



Quebra da propriedade de irmandade no nó P!

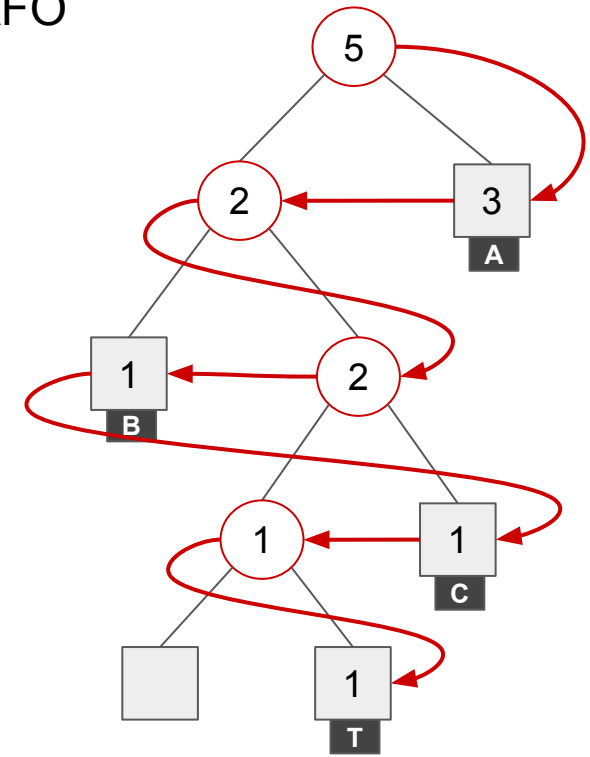
Quando isso ocorre, trocamos o nó com o líder do bloco de frequências ao qual ele pertencia antes do incremento!

Importante: a troca só não é realizada se o líder for ascendente de P

Atualiza nós

Exemplo

M = ABACA**T**EABAFABAFO

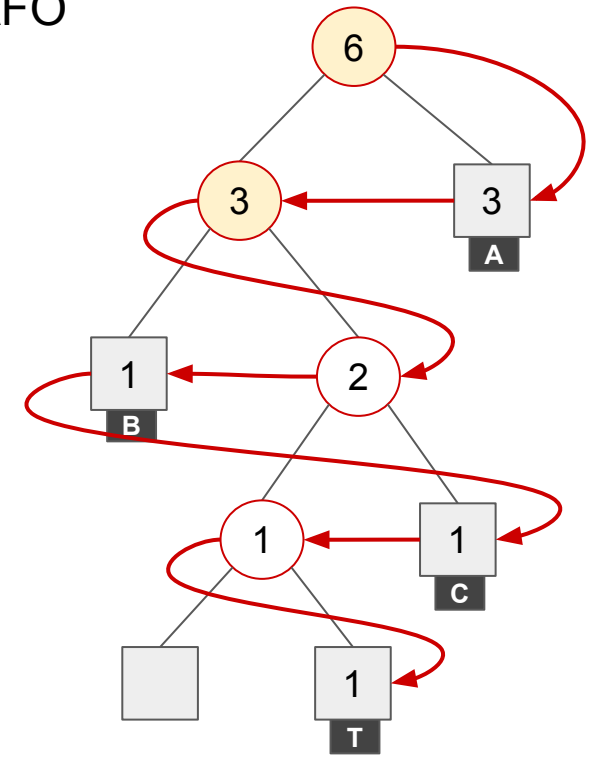


Quebra da propriedade de irmandade no nó P!

Efetua a troca

Exemplo

M = ABACA**T**EABAFABAFO

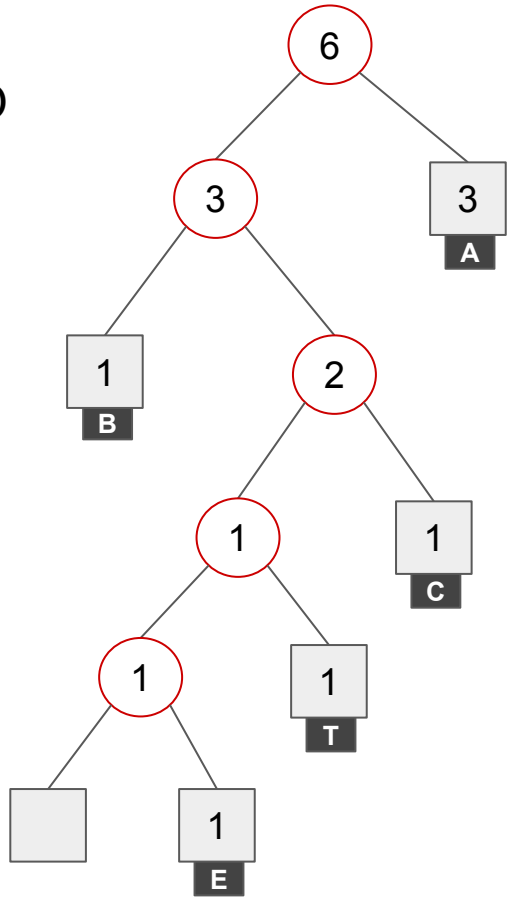


Propriedade de irmandade OK!

Continua a atualização dos nós

Exemplo

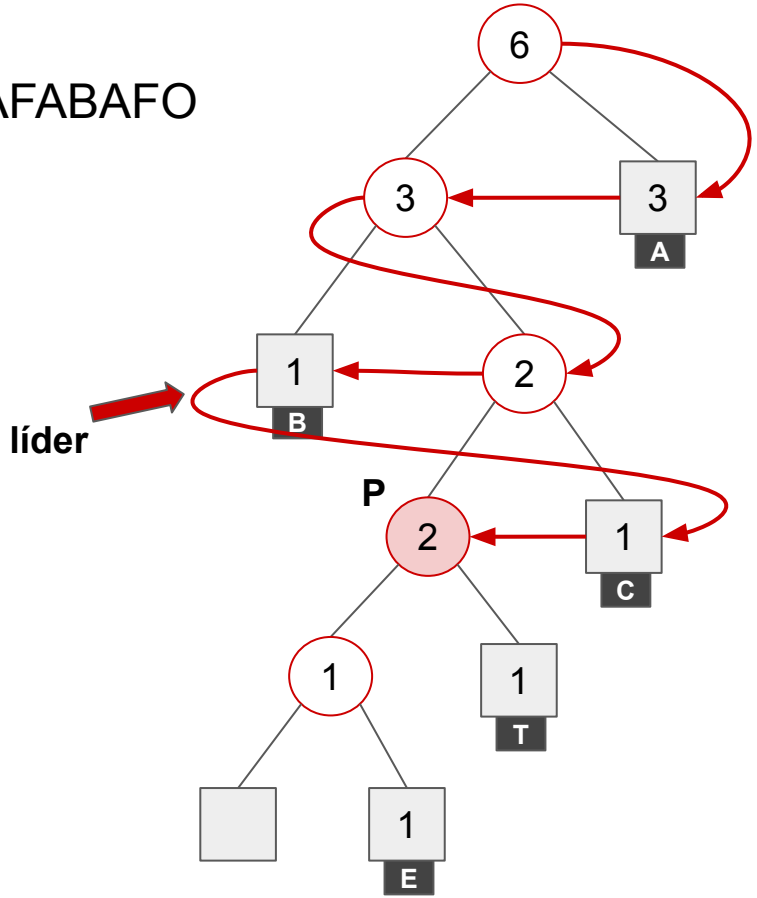
M = ABACAT**E**ABAFABAFO



Primeira ocorrência de E

Exemplo

M = ABACATEABAFABAFO

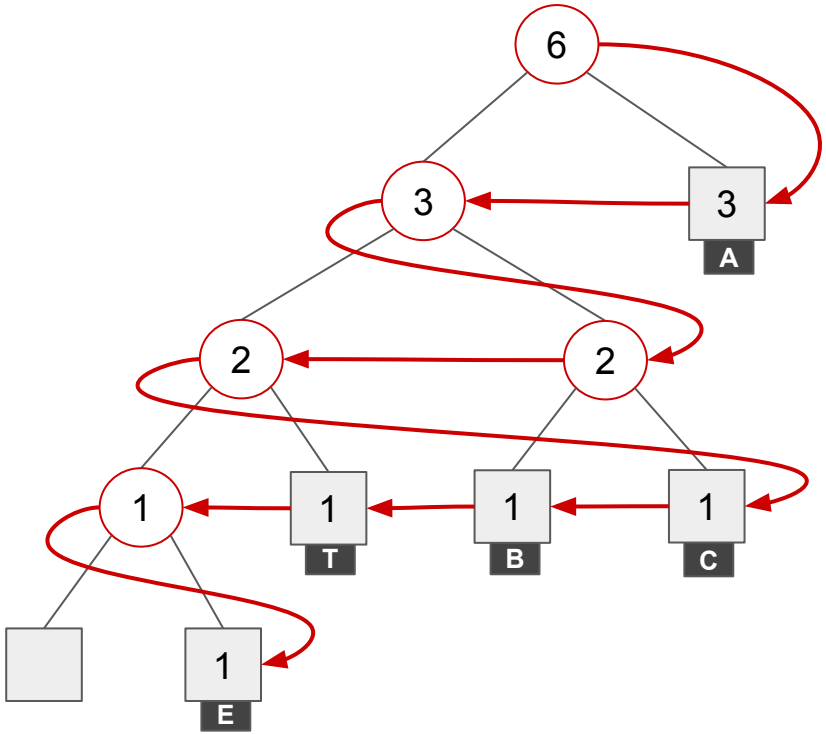


Quebra da propriedade de irmandade no nó P!

Atualiza nós

Exemplo

M = ABACATE**E**ABAFABAFO

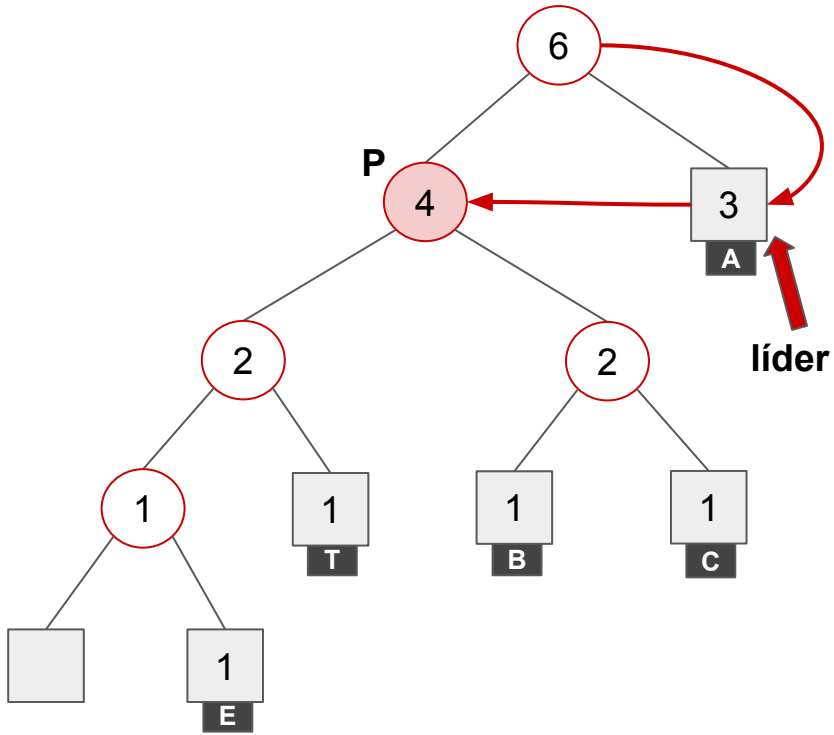


Quebra da propriedade de irmandade no nó P!

Efetua a troca

Exemplo

M = ABACAT**E**ABAFABAFO

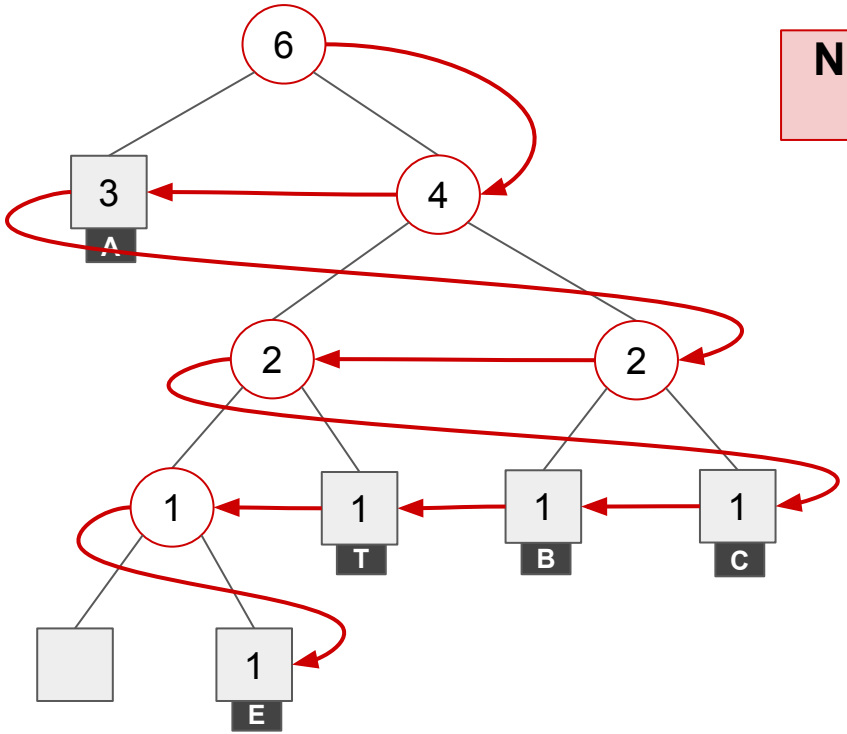


Nova quebra da propriedade de irmandade!

Continua a atualização dos nós

Exemplo

M = ABACAT**E**ABAFABAFO

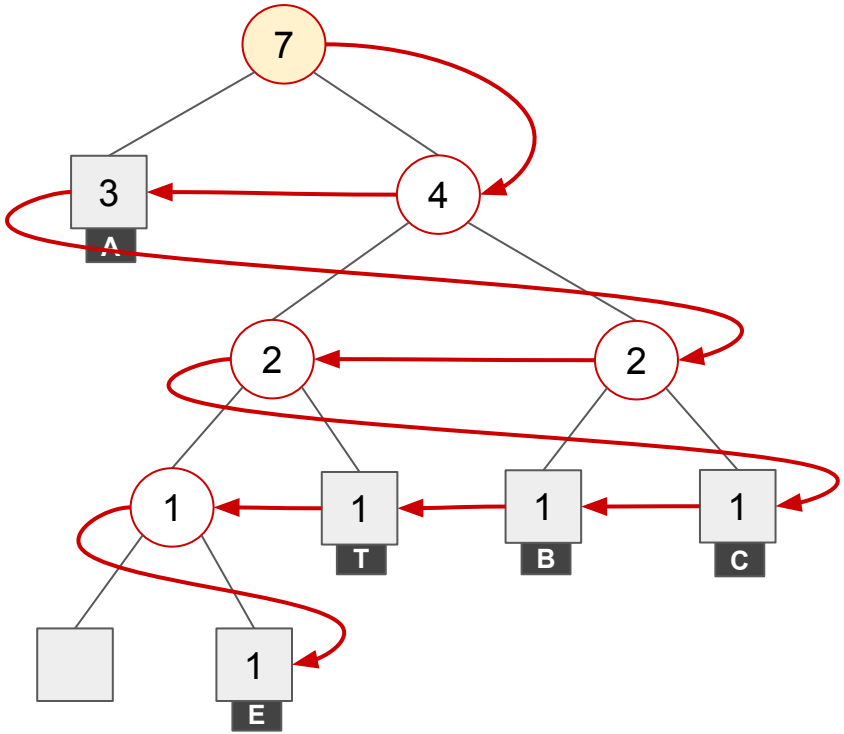


Nova quebra da propriedade de irmandade!

Efetua a troca

Exemplo

M = ABACATEABAFABAFO

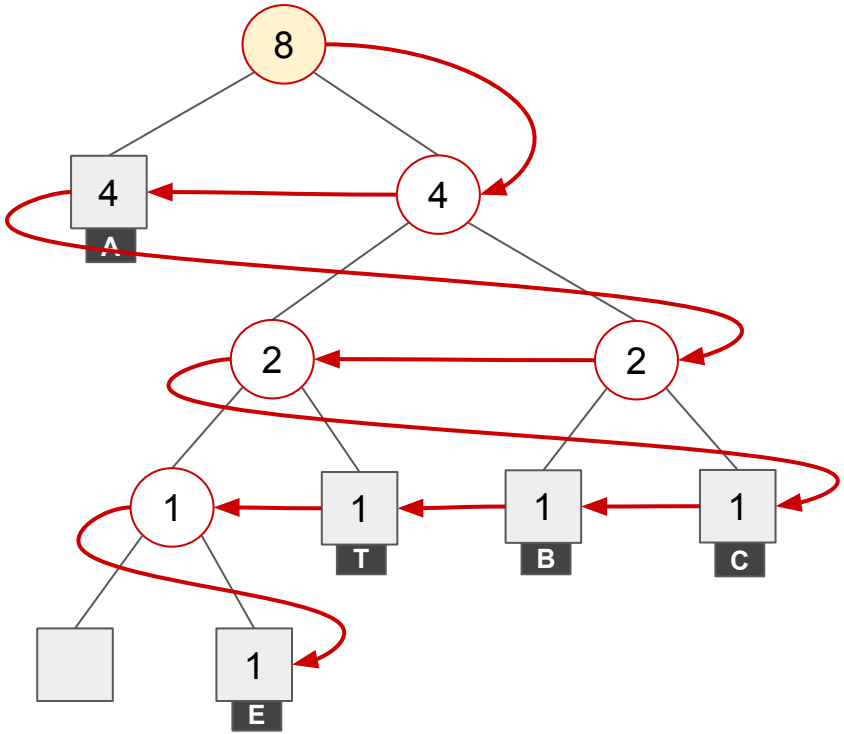


Propriedade de irmandade OK!

Continua a atualização dos nós

Exemplo

M = ABACATE**A**BAFABAFO

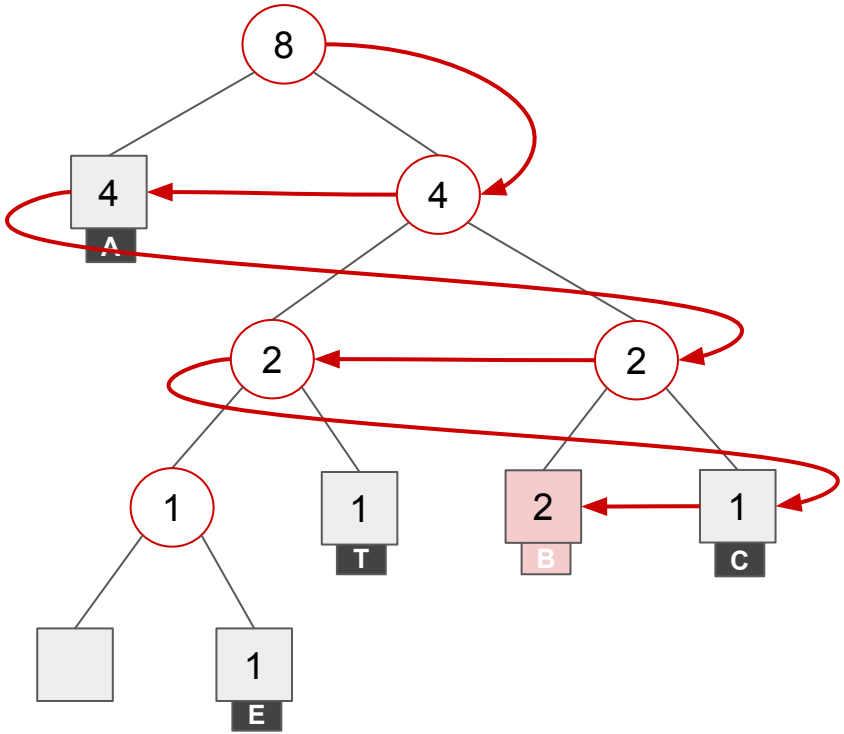


Propriedade de irmandade OK!

Quarta ocorrência de A, incrementa e atualiza

Exemplo

M = ABACATEA**B**AFABAFO

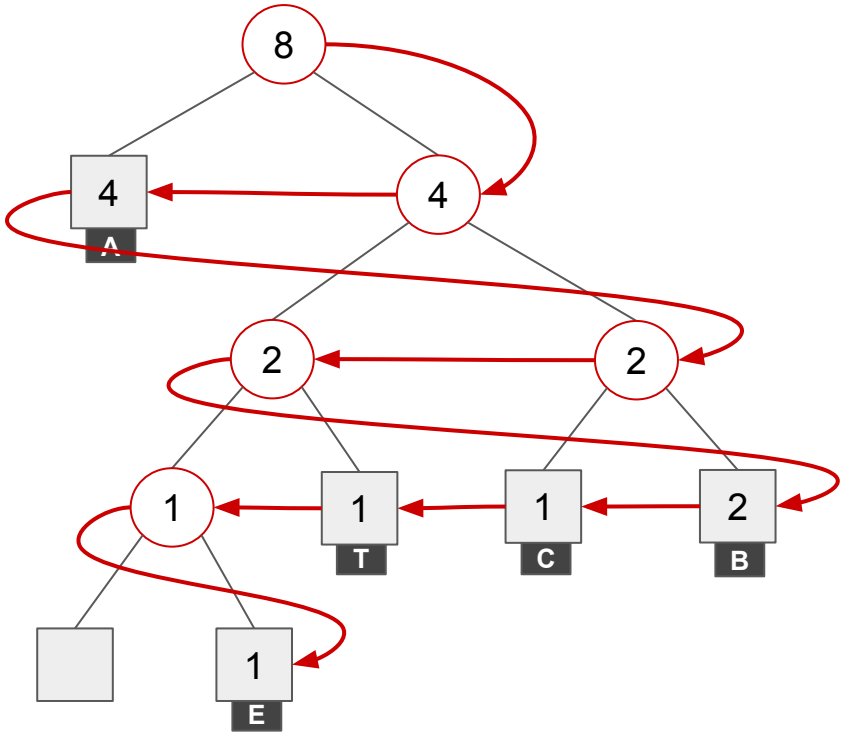


Quebra da propriedade de irmandade no nó P!

Segunda ocorrência de B, incrementa e atualiza

Exemplo

M = ABACATEA**B**AFABAFO

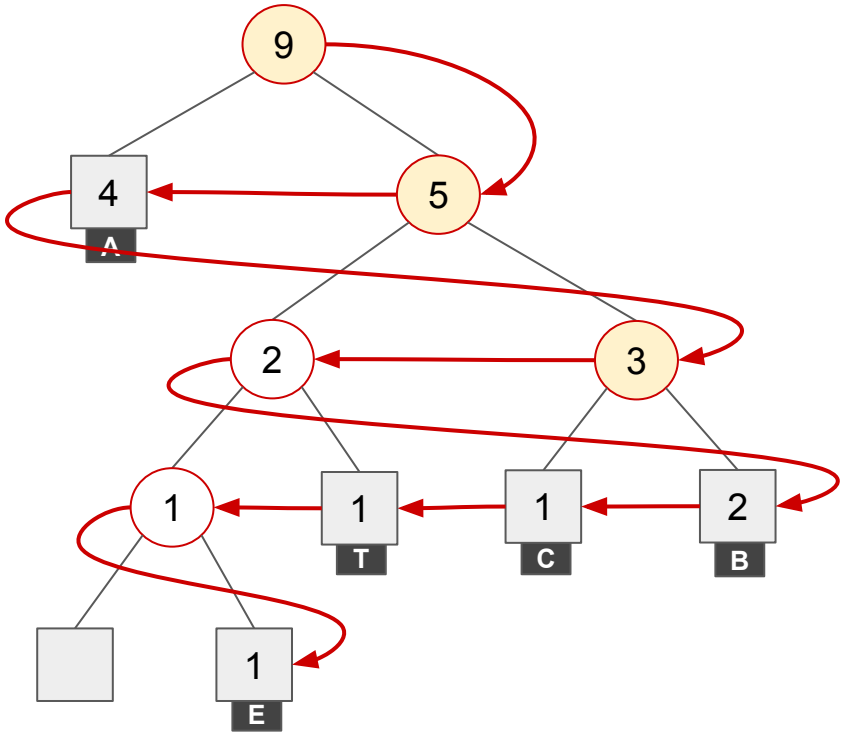


Quebra da propriedade de irmandade no nó P!

Efetua a troca

Exemplo

M = ABACATEA**B**AFABAFO

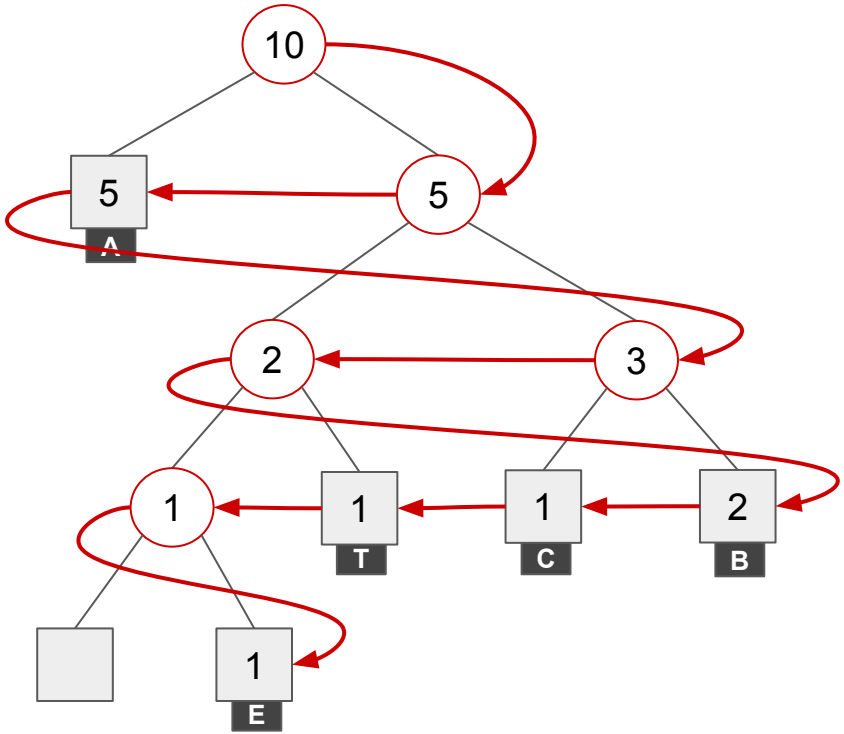


Propriedade de irmandade OK!

Continua a atualização dos nós

Exemplo

M = ABACATEABAFABAFO



Propriedade de irmandade OK!

E o processo continua até codificar toda a mensagem...

Quinta ocorrência de A, incrementa e atualiza

Huffman adaptativo

- E como fica a mensagem gerada?
  - Temos que olhar para o passo-a-passo do algoritmo
  - A mensagem é codificada simultaneamente à construção da árvore
  - Se um novo caractere é inserido, o código é o caminho do nó de frequência 0 seguido do caractere
  - Se o caractere já está presente na árvore, o código é o caminho para encontrá-lo **naquele momento**
    - Ou seja, o código para um mesmo caractere pode variar ao longo da codificação, à medida que sua frequência é atualizada
    - O processo de decodificação irá gerar exatamente a mesma árvore que foi gerada na codificação

Huffman adaptativo

- Vamos olhar somente a primeira parte da mensagem do exemplo anterior

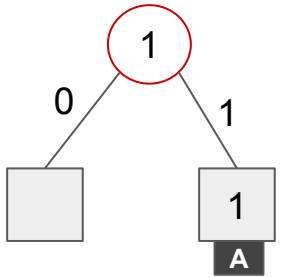
M = **A**BACATE
Saída: A

Árvore vazia: emite A

Huffman adaptativo

- Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABACATE
Saída: A0B

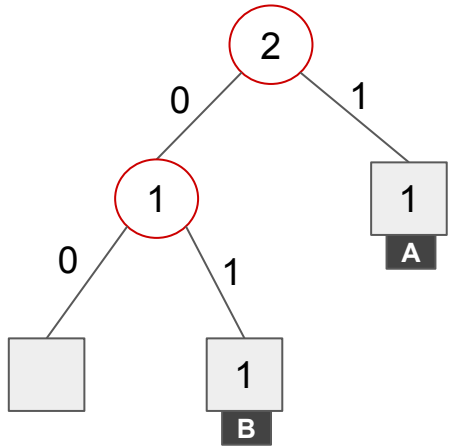


Inserção de novo caractere: caminho do nó 0 + caractere

Huffman adaptativo

- Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABACATE
Saída: A0B1

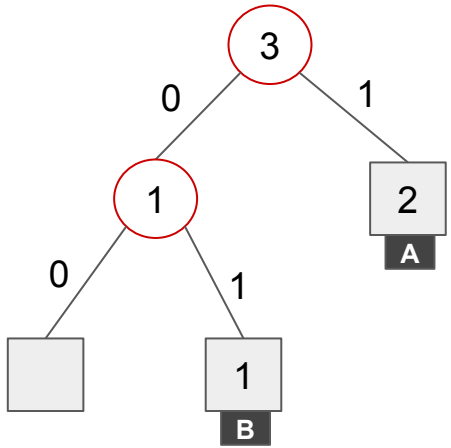


Nova ocorrência de A: caminho corrente até o caractere

Huffman adaptativo

- Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABA**C**ATE
Saída: A0B100C



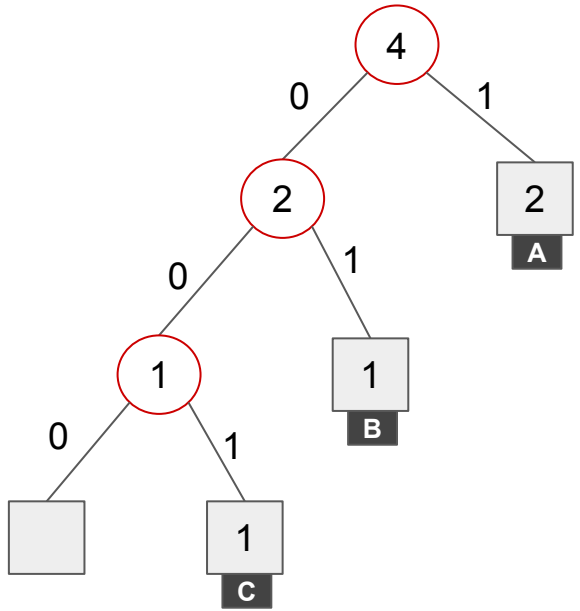
Inserção de novo caractere: caminho do nó 0 + caractere

Huffman adaptativo

➢ Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABACATE
Saída: A0B100C1

Nova ocorrência de A: caminho corrente até o caractere

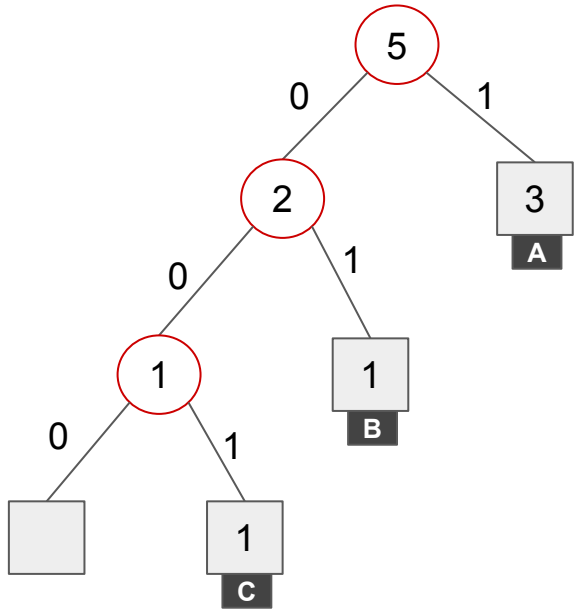


Huffman adaptativo

➢ Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABACA**T**E
Saída: A0B100C1000T

Inserção de novo caractere: caminho do nó 0 + caractere

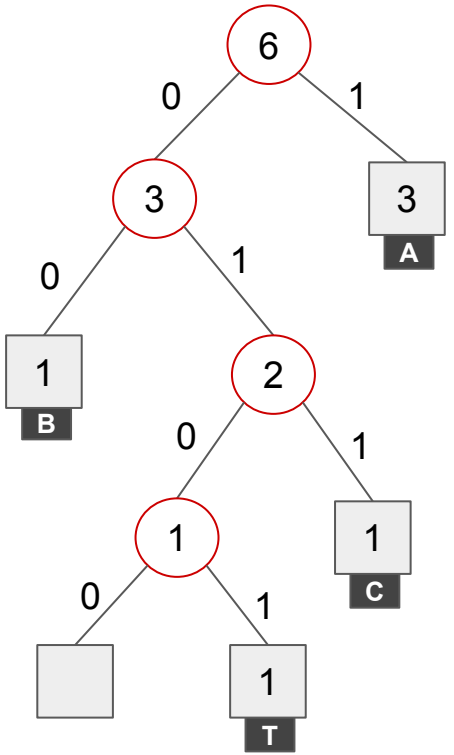


Huffman adaptativo

➢ Vamos olhar somente a primeira parte da mensagem do exemplo anterior

M = ABACATE
Saída: A0B100C1000T0100E

Inserção de novo caractere: caminho do nó 0 + caractere



Huffman adaptativo

ALGORITMO

```
p = nó folha que contém o caractere
codigo = sequência de bits até p
se (p == nó de frequência 0)
    codigo = codigo + caractere
enquanto (p não é raiz)
    i = frequência(p) - 1;
    se ((p viola a propriedade de irmandade) e
        (lider(i) não é ascendente(p)))
       troca(p, líder);
    p = ascendente(p);
    incremente frequência(p);
retorne codigo
```

Exercício 2

Use Huffman adaptativo para codificar o texto: **ABRACADABRA**

Referências

- DROZDEK, Adam. Data Structures and Algorithms in C++, Fourth Edition, cap. 11. Cengage Learning, 2013.
- CORMEN, T.; Leiserson, C.; Rivest, R; Stein, C. Introduction to Algorithms, Third Edition, cap. 16. MIT Press, 2009.
- SEDGEWICK, Robert. Algorithms, cap. 22. 1984.
- SOUZA, Jairo F. Notas de aula de Estrutura de Dados II. 2016. Disponível em: http://www.ufjf.br/jairo_souza/ensino/material/ed2/